移动平台开发书库



Android项目开发详解

翟大昆　陈春荼　编著

Android Project Development

- 内容安排由浅入深、逐步提高、循序渐进
- 本书结合实际开发项目，讲解开发流程和开发所用知识点
- 突出实用性，强调项目开发连续性





机械工业出版社
CHINA MACHINE PRESS

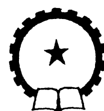
VISIT…

LANZAROTE Caliente.COM

移动平台开发书库

Android 项目开发详解

翟大昆　陈春茶　编著



机械工业出版社

本书从实战的角度出发，介绍了基于 Android 环境的 Mobile/Server 应用系统开发，涉及移动客户端和服务器端的开发环节，以满足智能手机在线应用的需求。全书以一个 Android 应用程序开发案例为主线，介绍了整个 Android 项目开发的过程。在内容的编排方面，为突出项目编程方法与编程思路这条主线，将 Android 组件的使用分配到每一个功能界面中讲解。本书涵盖了 Android 前端界面开发、Activity java 程序的开发以及 ASP 后台技术开发等内容，强调了 Android 与后台数据通信技术、XML 解析技术、数据存储技术以及与 Android 前端配套的 ASP 后台技术的实现细节。

每章的案例均有源程序，可登录 www.cmpbook.com 免费下载，以帮助读者深入地进行学习。

本书可作为高等院校相关专业学生学习嵌入式开发和手机开发的实训教材，也可供从事 Android 手机开发软件开发人员参考。

图书在版编目（CIP）数据

Android 项目开发详解 / 翟大昆等编著. —北京：机械工业出版社，2012.4
（移动平台开发书库）
ISBN 978-7-111-37968-3

Ⅰ. ①A…　Ⅱ. ①翟…　Ⅲ. ①移动终端－应用程序－程序设计
Ⅳ. ①TN929.53

中国版本图书馆 CIP 数据核字（2012）第 063092 号

机械工业出版社（北京市百万庄大街 22 号　邮政编码 100037）
责任编辑：郝建伟　曹文胜
责任印制：杨　曦

北京四季青印刷厂印刷

2012 年 6 月第 1 版 • 第 1 次印刷
184mm×260mm • 25 印张 • 616 千字
0001－3500 册
标准书号：ISBN 978-7-111-37968-3
定价：62.00 元

凡购本书，如有缺页、倒页、脱页，由本社发行部调换

电话服务
社服务中心：（010）88361066
销售一部：（010）68326294
销售二部：（010）88379649
读者购书热线：（010）88379203

网络服务
门户网：http://www.cmpbook.com
教材网：http://www.cmpedu.com

封面无防伪标均为盗版

前　言

目前介绍 Android 开发技术的书颇多，这些书的最大共同点就是花大量的篇幅介绍 Android 的各种开发控件的使用。的确，要掌握一门开发技术，要从基础控件的使用方法开始学起。不过，把这些控件罗列起来并未形成应用程序开发的一条主线，这会使初学者只了解了各个组件的使用，而对于 XML 解析、服务器端的数据存储等应用的核心环节不甚了解，难以进行 Mobile/Server 应用程序的整体性开发。这就好比，如果厨师把各种配料搅拌在一起，不恰当地调色调味，就不可能做出色、香、味齐全的菜肴。所有的软件开发，都是为了解决某一个系统性的问题，也就是 Mobile/Server 应用程序。然而信息处理系统本身还是有一定的规律可循，信息处理系统如果按功能划分，不外乎是输入、输出、增、删、改、查询、分类等内容。在 Mobile/Server 应用程序开发过程中，就是要快速地将应用程序分解成各个功能模块，然后将各个功能模块再次分解成各个功能组件。因此，在程序开发的学习过程中，就要加强 Mobile/Server 应用程序的实战训练，等到真正要开发实际 Mobile/Server 应用程序的时候，才能运筹帷幄，各个组件使用自如。还是拿厨师做菜比喻，厨师要将适合各种菜肴的配料准备齐全，在学习炒菜的时候就要研究各种菜肴的做法，而不是等到客人点这道菜的时候才研究做法。

有过 Web 项目开发经验的人，都会有这样的感受，Web 项目开发存在着某一套模板，有几个共同的模块是通常都有的，比如登录、注册及具体功能模块。Android 应用程序项目的开发也是如此，存在着某种规律可供读者学习。本书的特点是：站在项目开发的角度学习 Android 手机开发，除使用经典的示例讲述 Android 基础控件的使用外，还强调了 Android 的 Mobile/Server 应用程序开发的整体性，系统地介绍了项目开发的过程。本书以一个手机订票系统项目为主线，分功能模块来介绍 Android 项目开发的过程，对于每个功能模块又分别介绍其所涉及的控件及实现该功能模块需要掌握的重点知识。当各个模块功能都实现了，也就完成了项目开发。使用一个具体的案例项目来描述 Android 项目开发，旨在抛砖引玉，虽然每个项目都有其具体性及局限性，但目的是通过一个典型案例，使读者熟悉 Android 项目开发的流程，了解各种控件的使用，并将这些控件的功能经过取舍、再组装运用于其他项目的开发中。本书还涉及了 Android 前台如何与数据后台进行通信，恰当地组装数据流便于数据的解析与使用等内容，这些内容都是开发每一个项目必不可少的核心技术，如果读者掌握了这一开发过程，相信在今后的实际项目开发过程中定会信心十足，游刃有余。

本书不采用罗列 Android 所有组件使用方法的编程风格，而是结合每一个功能界面所涉及的控件，在每一章的“基础控件讲解”部分讲解，需要重点掌握的 Android 编程方法与编程思路，则在每一章的“重点剖析”里讲解。读者可在学会了基础控件的使用以及 Android 编程思路等方法后，再进行项目开发。本书中给出的一个案例项目，是供读者学习和参考的，读者也可以根据自己的兴趣爱好，换成其他项目，达到举一反三的效果。所以本书给读

者留出了足够的技术提升空间，以便真正掌握 Android 这门手机开发技术。这样编排的目的是为了突出实用性，强调项目开发的连贯性，读者无需一下子花大量的时间，一次性学习所有控件的使用方法，而是根据“功能界面的需要”进行学习。这样做既能强调控件如何在一个整体项目中具体运用，又有利于读者循序渐进地进行 Android 学习。本书中给出的案例项目基本覆盖了所有基础控件的使用。

本书不涉及游戏开发，如果读者想了解 Android 游戏开发部分的知识，可以参考相关的书籍。之所以不讲述这方面的知识，原因有两个：一是初学 Android 的读者掌握游戏开发有一定难度，二是游戏开发不仅是技术问题，更重要的是游戏创意。

本书贯彻学以致用的思想，在内容安排上由浅入深，逐步提高，强调循序渐进的过程。

如果读者是一名熟练的程序员，那么本书是 Android 手机编程世界中的里程碑，因为本书就是程序员亲历亲为的项目总结，且本书的编排也和平时项目开发的过程相同。

如果读者对程序设计还不熟悉，没有程序开发经验，那么通过本书可以掌握 Android 手机编程的基本技术以及项目开发思路。

如果读者是一名已掌握 ASP 开发，并对 Android 手机开发有浓厚兴趣的软件工程师，相信通过学习此书，会在技术上有突破，并增加就业与择业的砝码。

无论是对手机编程感兴趣的初学者，还是手机应用开发的工程师，本书都是使用和学习 Android 手机编程的最佳选择。

本书由翟大昆、陈春荼编著。张蕾、朱谨、张飞、杜树刚、程晓荣等对书稿内容提出了许多宝贵意见、给作者提供了很多支持和帮助，在此表示感谢。在编写过程中，我们参考和引用了一些国内外文献，在此谨向有关作者表示真诚的感谢。

为了方便读者学习，本书提供了大量源程序代码，读者可登录 www.cmpbook.com 免费下载。

由于时间仓促，书中难免存在不妥之处，请读者原谅，并提出宝贵意见。

编　者

目　　录

前言
第 1 章　Android 开发平台的构建…………1
1.1　Android 开发行业发展前景…………1
1.1.1　Android 概述…………1
1.1.2　Android 行业需求前景…………1
1.1.3　Android 就业前景…………2
1.2　开发平台的搭建…………3
1.3　体验 Android 开发之旅…………13
1.3.1　创建 HelloAndroid 项目…………13
1.3.2　模拟器配置及运行 HelloAndroid…………15
1.3.3　剖析 HelloAndroid 项目结构…………18
1.4　设置 Android 模拟器中文环境…………26
1.5　关于本书…………27
第 2 章　Android 项目开发——以手机订票系统为例…………30
2.1　项目需求概述…………30
2.2　项目功能分解…………30
2.2.1　程序主界面功能…………30
2.2.2　登录功能…………31
2.2.3　注册功能…………31
2.2.4　选择功能…………32
2.2.5　查询功能…………32
2.2.6　信息列表功能…………33
2.2.7　List 详情功能…………34
2.2.8　用户信息填写与选择功能…………34
2.2.9　预订成功功能…………35
2.3　项目后台技术选择——ASP.NET…………35
2.3.1　ASP.NET 网页特点…………36
2.3.2　ASP.NET 文件的体系结构…………36
2.3.3　ASP.NET 的工作原理…………37
2.3.4　IIS 的安装与配置…………37

第 3 章　程序主界面……38
3.1　基础控件讲解……38
3.1.1　网格视图……38
3.1.2　提示……42
3.2　重点剖析……45
3.2.1　数据适配器……45
3.2.2　Android 事件处理机制……49
3.2.3　Android 页面切换……54
3.3　程序主界面功能实现……58
3.3.1　程序主界面 View 实现……59
3.3.2　程序主界面 Control 实现……59
3.3.3　程序主界面 Help 实现……64
第 4 章　登录功能实现……66
4.1　基础控件讲解……66
4.1.1　文本框……66
4.1.2　文本编辑框……69
4.1.3　单选按钮……71
4.1.4　按钮控件……74
4.2　重点剖析……76
4.2.1　Android 长度单位……76
4.2.2　Android 布局的用法……78
4.2.3　SharedPreferences 的使用……91
4.2.4　Android 解析 XML 的方法……95
4.2.5　Android 如何与后台通信……113
4.3　登录功能实现……143
4.3.1　登录界面 View 实现……143
4.3.2　登录功能 Model 用户信息类实现……146
4.3.3　登录功能 Control 实现……147
4.3.4　登录功能 Help 实现……148
4.3.5　登录后台 ASP 实现……148
第 5 章　注册功能实现……150
5.1　重点剖析……150
5.1.1　Activity 页面传值……150
5.1.2　使用正则表达式进行信息验证……154
5.2　注册功能实现……162
5.2.1　注册界面 View 实现……162
5.2.2　注册功能 Control 实现……165

5.2.3 注册后台 ASP 实现 ············166
第 6 章 选择功能实现 ············167
6.1 基础控件讲解 ············167
6.1.1 自动匹配 ············167
6.1.2 列表视图 ············169
6.2 重点剖析 ············172
6.2.1 ListAdapter+HashMap+ArrayList 的使用 ············172
6.2.2 SimpleCursorAdapter 的使用 ············175
6.3 选择功能实现 ············179
6.3.1 选择界面 View 实现 ············179
6.3.2 选择功能 Model 类实现 ············180
6.3.3 选择功能 Control 实现 ············181
6.3.4 选择功能后台 ASP 实现 ············181
第 7 章 查询功能实现 ············183
7.1 基础控件讲解 ············183
7.1.1 图片视图 ············183
7.1.2 图片按钮 ············186
7.1.3 下拉列表 ············189
7.1.4 日期和时间控件 ············192
7.1.5 菜单 ············196
7.1.6 对话框 ············203
7.1.7 进度条 ············214
7.2 重点剖析 ············217
7.3 查询功能实现 ············222
7.3.1 查询界面 View 实现 ············222
7.3.2 查询功能 Control 实现 ············226
7.3.3 查询功能 ASP 实现 ············226
第 8 章 信息列表功能实现 ············229
8.1 重点剖析 ············229
8.1.1 日期格式处理与转换 ············229
8.1.2 带图片多行 ListView 子项 ············233
8.2 信息列表功能实现 ············237
8.2.1 信息列表功能 View 实现 ············237
8.2.2 信息列表功能 Model 类实现 ············241
8.2.3 信息列表功能 Control 实现 ············246
第 9 章 List 详情功能实现 ············248
9.1 基础控件讲解 ············248

9.1.1　文字交换器 ······248
9.1.2　图片交换器 ······251
9.2　List 详情功能实现 ······253
9.2.1　List 详情界面 View 实现 ······253
9.2.2　List 详情功能 Control 实现 ······258
第 10 章　用户信息选择与填写功能实现 ······259
10.1　基础控件讲解 ······259
10.1.1　复选框 ······259
10.1.2　卷轴视图 ······264
10.2　重点剖析 ······267
10.2.1　如何动态增加控件 ······267
10.2.2　多行文本滚动实现 ······270
10.2.3　用户身份证的验证 ······272
10.3　用户信息选择与填写（机票预订）实现 ······276
10.3.1　机票预订 View 实现 ······276
10.3.2　机票预订 Model 实现 ······280
10.3.3　机票预订功能 Control 实现 ······280
10.3.4　机票预订后台 ASP 实现 ······281
第 11 章　订购成功实现 ······282
11.1　基础控件讲解 ······282
11.1.1　状态栏提示 ······282
11.1.2　拖动条 ······285
11.1.3　循环播放图片列表效果 ······288
11.2　订购成功实现 ······293
11.2.1　机票订购成功 View 实现 ······293
11.2.2　机票订购成功功能 Control 实现 ······294
第 12 章　界面优化、程序发布与真机环境测试 ······296
12.1　界面效果优化 ······296
12.2　程序自适应处理 ······298
12.3　设置程序 Logo ······303
12.4　Android 单元测试 ······303
12.5　项目在不同版本的 Android 手机上的安装与运行 ······310
第 13 章　其他 Andorid 专题开发 ······311
13.1　专题一：Andorid 数据存储 ······311
13.1.1　Files 存储 ······311
13.1.2　NetWork 存储 ······317
13.1.3　SQLite 编程详解 ······322

13.1.4 Content Providers 数据存储 ··· 330
13.2 专题二：Android 多媒体开发 ··· 344
13.2.1 播放音乐 ··· 344
13.2.2 播放视频 ··· 350
13.3 专题三：外部接口编程 ··· 353
附录 ··· 372
附录 A IIS 部署及网站发布 ··· 372
附录 B 91 助手的安装与使用 ··· 378
附录 C DDMS 常用功能详解 ··· 384
参考文献 ··· 388

第 1 章　Android开发平台的构建

本章是 Android 应用开发的预备知识导读。介绍了 Android 开发的行业发展前景及就业前景。讲解了如何搭建与配置 Android 开发平台，详细地剖析了第一个 Android 程序的结构，并就如何设置 Android 模拟器中文环境做了总结，旨在让读者尽快地了解 Android 应用程序的结构和开发平台设置等相关知识，本章的最后一节是关于全书内容的总体描述。

1.1　Android开发行业发展前景

近年来在开放的手持设备中 Android 无疑是发展最快的操作系统之一，Android 产品上市速度快，体现了 Google 移动平台的强势。

1.1.1　Android概述

Android 有两层意思，本义是指“机器人”，另一层意思是指 Google 于 2007 年 11 月 5 日发布的基于 Linux 核心的开源手机软件平台，该平台主要由操作系统、中间件、用户界面和应用软件组成，Google 官方称它是第一个为移动终端打造的真正开放和完整的移动软件平台。

目前应用 Android 平台的人员主要分为两类，一类是设备制造商，他们希望在自己的硬件平台上运行 Android；另外一类人员是移动应用开发者，希望将自己的应用移植到该平台上或者在该平台上开发相关应用。本书主要讨论的是 Android 应用软件开发，这是因为应用软件开发在移动市场上的前景是最广阔的，也是门槛最低的。目前智能手机的应用已经越来越广泛，市场上已经出现数万种运行于 Android 平台的手机应用软件，包括办公软件、影视娱乐软件、游戏软件、具体功能软件等应用领域，可以说已深入到移动应用的方方面面。应用软件开发人才的需求数量庞大，据统计，软件应用类 Android 开发人才的需求约占总需求的 72%。基于这两点原因，本书作者想把自己积累的 Android 项目开发经验及开发心得与读者分享。

1.1.2　Android行业需求前景

当前计算机技术已经发展到了移动互联网时代，这意味着将有超过 10 亿人的智能设备，将随时随地接入网络之中进行信息交互，而移动互联网终端将成为人们获取信息和分享知识的重要窗口。移动终端的创新空间比起它的前辈 PC、PDA 来说大得无法想象，并且创新的速度正在加快，包括多核技术、并行计算、高清显示、3D 显示、多屏和变屏技术、多点触控、环境感应等新技术，以及运营商的越来越快的移动宽带互联网等，使移动终端成为推动移动互联网发展的强劲动力。伴随着移动互联网随时获取和分享知识重要性的增加，服务模式也需要创新。比

如，终端需要随时随地获得在线升级、安全以及数据与信息备份服务，同时用户可以随时体验最新的业务与应用。所有这些需求的满足，都要求终端企业把被动服务变成主动服务，把远距离的后台服务变为零距离的直观式体验服务。2011 年 7 月 26 日由奇虎 360 总裁齐向东提出的“得移动终端者得天下”的口号，预示着移动互联网的机会已经来临，于是将有更多的年轻人投身于移动互联网的创业大潮。

在最近的一年中，计算机右下角的战争烽烟未灭，移动互联网入口之争战火重燃。与计算机右下角相比，此次争夺的是被誉为“规模 10 倍于 PC 互联网”的移动互联网入口——手机操作系统，统称为“手机 OS”。中国互联网行业最活跃的人物“马云、李彦宏、马化腾、李开复、雷军” 都在为之疯狂，由于 Android 操作系统大势所趋、开放，但不够本土化，中国互联网公司从中看到了机会。阿里巴巴、百度、腾讯等互联网巨头皆在试水手机操作系统。同时，市场上还有两股不可忽视的力量：创新工场投资的点心 OS 和雷军创办的小米科技。这使我们不由得想起 PC 初始年代的操作系统之争，互联网初始年代的浏览器之争。最后真正主导信息系统发展的主要动力还是应用软件。应用软件的研发，有些是面向行业的，有些是面向企业的，有些甚至是面向个人的，范围之广，是一两个“云”平台难以覆盖的。所以需要一大批新生代应用软件研发人员。PC 初始年代的应用程序的开发历程也是从单机的应用程序开始，逐步过渡到 Client/Server、Browser/Server 的模式。目前的 Android 的应用程序开发大多停留在“从手机到手机”阶段，但随着移动互联网技术的普及，Mobile/Server 模式过渡的时间不会再像 PC 那样的缓慢。所以，手机应用软件的开发人员将要面对的是手机本身的应用程序技术和手机与移动互联网技术整合的复杂局面。

既然计算机已经逐步迈进了移动开发的大门，Android 又是移动开发的领军技术，下面让我们看看国内外 Android 的发展前景，以便更好地了解移动开发市场的动向。国外 Android 市场正在迅猛扩展。市场调研机构 NPD Group 最近发布的一份报告称，2011 年第一季度基于 Android 操作系统的智能手机，在美国智能手机总销量中所占比例达到 28%，首度超过苹果 iPhone，相信在不久的将来会有更多的用户选择 Android 系统的手机或是无线终端设备。目前中国市场上拥有世界上最大的手机用户群，再加上 3G 的推出对整个行业的巨大推动作用，全世界所有手机制造商几乎都在招聘 Android 工程师。因此我们可以预言，Android 在中国市场上的应用前景十分广阔。中国有成熟的消费群，Android 社区十分红火，这些社区为 Android 在中国的普及做出了较好的推广作用。国内厂商和运营商也纷纷加入了 Android 阵营，包括中国移动，中国联通，中兴通信，华为通信，联想等大企业。不仅仅局限于手机，国内厂家也陆续推出了采用 Android 系统的 MID 产品，比较有名的包括由 Rock Chip 和蓝魔共同推出的同时具备高清播放和智能系统的音悦汇 W7 手机和 2010 年推出的原道 N5 手机，可以预见 Android 也将会被广泛应用在国产智能上网设备上，这将进一步扩大 Android 系统的应用范围。

1.1.3 Android就业前景

目前对 Android 人才的需求主要集中于两类，一类是偏向硬件驱动的 Android 人才需求，另一类是偏向软件应用的 Android 人才需求。从目前的招聘需求来看，后者的需求最大，包括手机游戏、手机终端应用软件和其他手机应用软件的开发。据职酷网职位搜索库统



计显示，目前企业对这类应用软件的 Android 开发人才需求占总需求的 72%。据职业专家分析，由于目前 Android 技术较新，无论是相关书籍、培训还是大学教育，都处于初级阶段，因此 Android 人才短期将供不应求。从长期来看，随着各种移动应用和手机游戏等内容需求日益增加，也将激励大、中、小型手机应用开发商加大对 Android 应用的开发力度，因此 Android 人才的就业前景也非常可观。

据市场推断，伴随着移动市场的不断推进，3G 人才全球相对紧缺，实用性人才培养已迫在眉睫！在中国，三大运营商如火如荼的 3G 营销战持续升温，这对整个行业的发展具有巨大的推动作用，无疑将引爆 3G 手机开发工程师这个 IT 行业的高薪职位。所以，程序员必学 Google Android 的理由可以总结为以下几点：更快的薪酬提升通道、更好的热门就业岗位、更多的行业人才需求、最热门的新技术行业。普通软件工程师与 Android 软件工程师的薪资区别如图 1-1 所示。

据职酷网数据统计显示：目前热招的 Android 技术相关岗位约有 3882 个，而一个月内的有效岗位量在 2298 个，目前 Android 热招的岗位如图 1-2 所示。

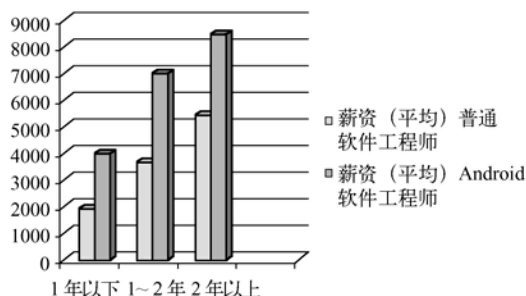


图 1-1　普通软件工程师与 Android 软件工程师薪资对比

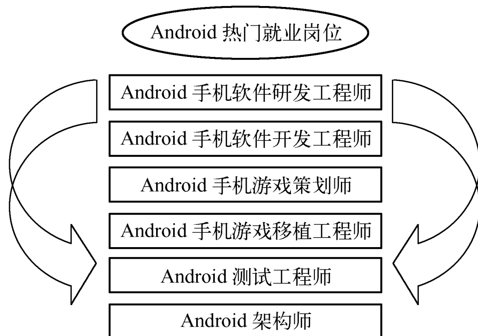


图 1-2　Android 热门就业岗位

1.2　开发平台的搭建

看到上面介绍的 Android 行业前景及就业前景，相信读者都想尽快掌握 Android 开发技术。“工欲善其事，必先利其器”我们首先得学习如何搭建 Android 开发平台。搭建 Android 开发平台涉及的开发工具有：JDK（java 虚拟机）、Eclipse 开发工具、Android SDK 及 ADT。下面讲述这几款软件的下载地址及在 Android 开发中扮演的功能角色，如表 1-1 和表 1-2 所示。

表 1-1　Android 开发所需软件的下载地址

软 件 名 称	下 载 地 址
JDK	http：//java.sun.com
Eclipse	http：//www.eclipse.org
Android SDK	http：//developer.android.com/sdk/index.html
ADT	https：//dl-ssl.google.com/android/eclipse/（可在线安装，不必下载）

表 1-2　Android 开发所需软件在项目开发中的功能介绍

软 件 名 称	在 Android 项目开发中的功能角色
JDK	Android 开发是以 Java 作为开发语言，所有用 Java 开发的应用程序都需要安装 Java 虚拟机
Eclipse	Eclipse 是一款免费、优秀开源的集成开发平台（IDE），很多 Java 项目开发都是基于这个平台，Android 应用程序开发也不例外
Android SDK	Android 软件开发工具包，是应用软件开发工具的集合。该工具包定义了很多 Android 手机开发的底层应用，可以调用这些底层工具实现更多、更复杂的手机应用
ADT	将 Eclipse 和 Android SDK 连接起来的纽带，在 Eclipse 编译 IDE 环境中，安装 ADT，为 Android 开发提供开发工具的升级或者变更

下面重点讲述 Android 开发平台的搭建步骤：

第一步，安装 Java SDK 并配置 Java 开发环境。

第二步，Eclipse 开发工具安装与汉化。

第三步，Android SDK 安装与配置（全部安装有点慢，需耐心等候）。

第四步，ADT 安装与配置。

（1）安装 Java SDK 并配置 Java 开发环境

首先安装 java 虚拟机。单击 JDK，安装过程中所有选项保持默认（路径为：C:\Program Files\Java）并按顺序安装即可。

其次，配置 JDK 的环境变量：在“我的电脑（Windows 7 是计算机）”图标上单击右键→“属性”→“高级”→ “环境变量(N)”，如图 1-3 所示。

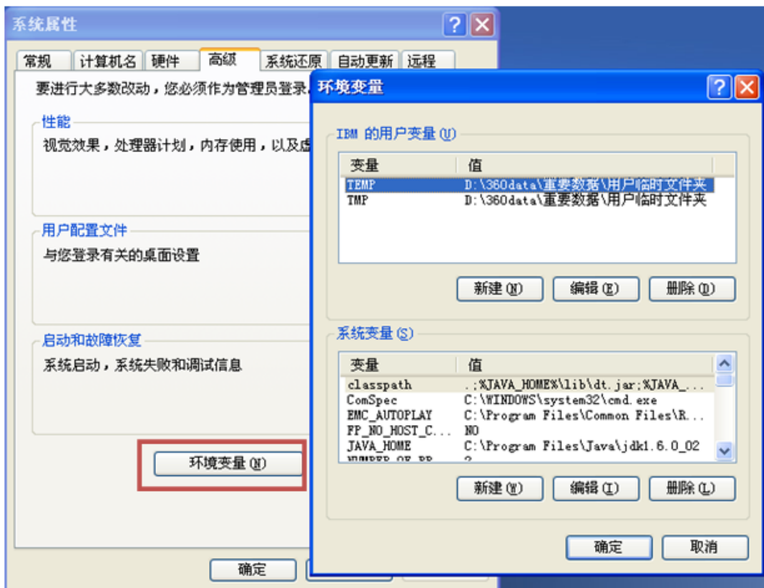


图 1-3　JDK 环境配置位置

配置 path 变量：在系统变量（S）中找到 path，一般已经有 path 变量，若 path 中有其他值，不要把 path 中的值删除，只要把 path 相应的值添加进去就行。如果没有，则新建 path 变量。单击“编辑”按钮，将 Java JDK 安装目录中的“bin”文件夹路径添加到 path 变量值中，如：变量值（V）：C:\Program Files\Java\jdk1.6.0_02\bin;单击“确定”按钮完成，如图 1-4 所示。



配置 classpath：在系统变量（S）中找到 classpath，一般已经有，如果没有新建 classpath 变量。单击“编辑”按钮，将 Java JDK 安装目录中的“lib”文件夹路径添加到 classpath 变量值中，如：变量值（V）：.;%JAVA_HOME%\lib\dt.jar;%JAVA_HOME%\lib\tools.jar（.;一定不能少，因为它代表当前路径），单击“确定”按钮完成，如图 1-5 所示。

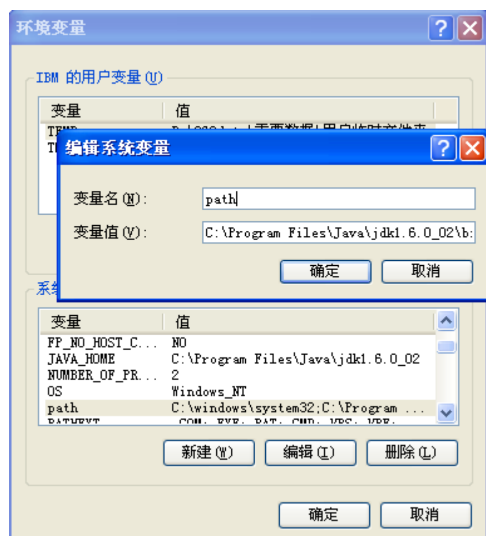


图 1-4　配置 path 变量

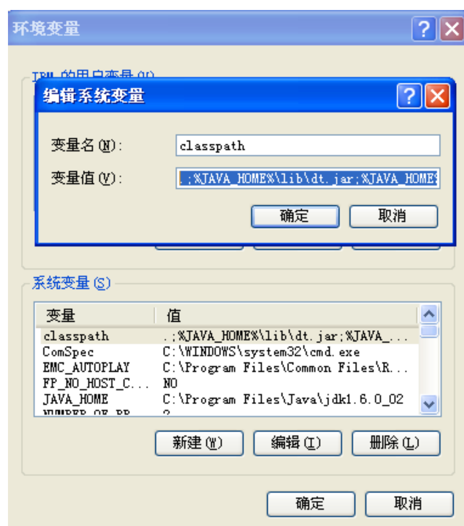


图 1-5　配置 classpath 变量

用下面的方法测试 JDK 是否安装成功：

单击“开始”菜单→“运行”→“CMD”，在 MS-DOS 中输入：“java –version”（能看到当前安装的 JDK 的版本号），再输入：JAVA，JAVAC，此时可以看到很多参数的提示，如图 1-6 所示。

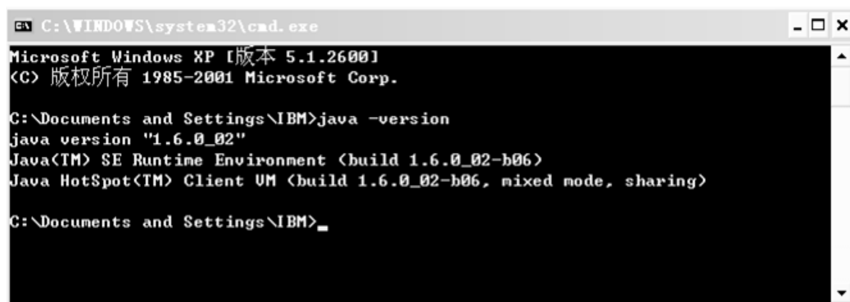


图 1-6　Java SDK 安装成功

（2）Eclipse 开发工具的安装与汉化

Eclipse 的安装很简单，将其解压即可，记住解压的路径。如：D:\Program Files\eclipse-SDK-3.6.2-win32\eclipse。本书使用的是 3.6.2 版本的 Eclipse，Eclipse 最近发布了解决国际化问题的独立语言包，也就是如何将 Eclipse 汉化的问题。具体步骤如下：

1）启动 Eclipse 开发工具，选择工作区，如图 1-7 所示。依次单击菜单栏中的“Help”→“Software Update…”（3.6.2 版本是“Install New Software...”）菜单命令，如图 1-8 所示。

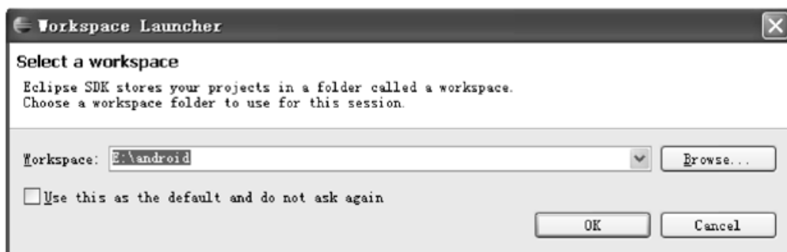


图 1-7　Eclipse 开发工作区

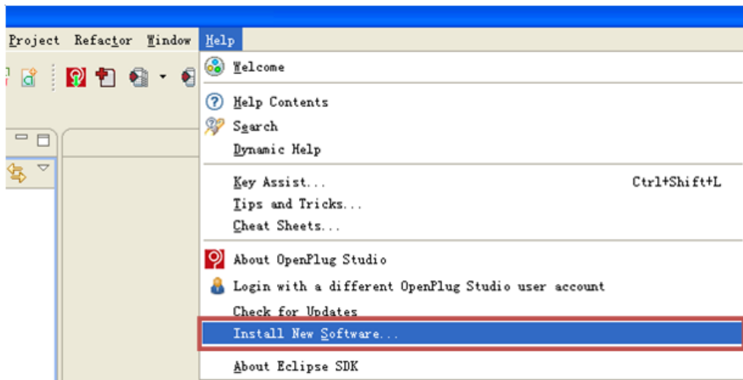


图 1-8　菜单栏中 Help→Install New Software...的位置

2）打开对话框，选择“Add”项，弹出需要下载的汉化包地址，如图 1-9 所示。不同的 Eclipse 版本汉化包地址是不一样的，本书使用 3.6 版本，地址是：http://archive.eclipse.org/technology/babel/update-site/R0.8.0/helios。给路径取一个别名，然后在 Location 中输入地址，如图 1-10 所示。然后单击“OK”按钮，经过下载，在 Name 子项中会出现许多项，选择简体中文，如图 1-11 所示。

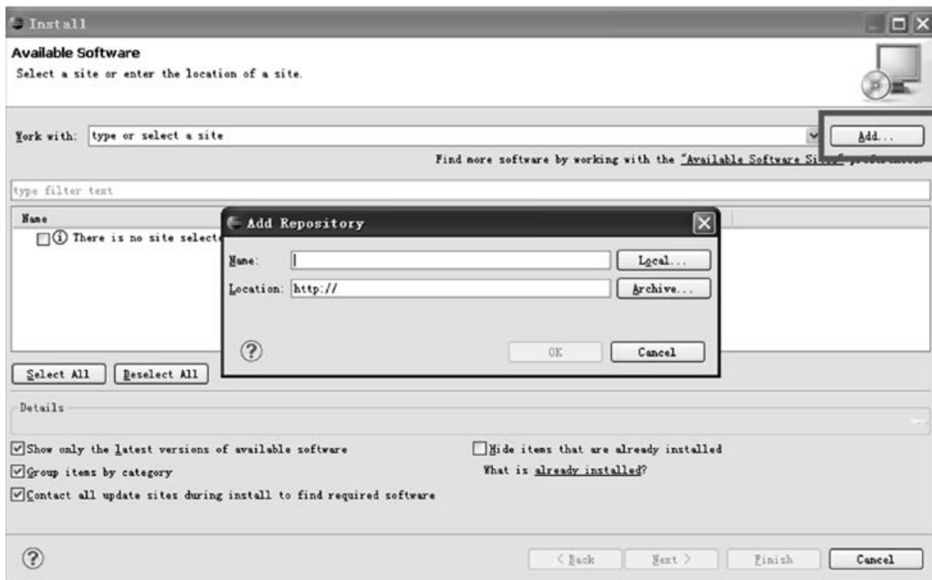


图 1-9　添加汉化包地址（1）



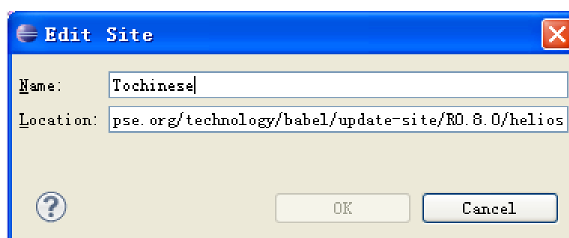


图 1-10　添加汉化包地址（2）

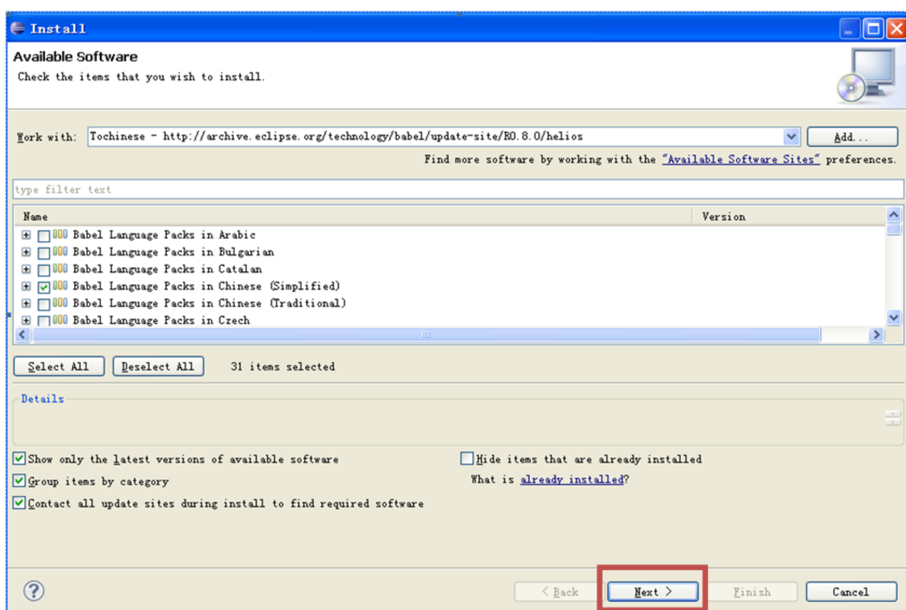


图 1-11　选择简体中文汉化包

3）单击“Next”按钮，直到出现如图 1-12 所示界面，选择“I accept the terms of the license agreement”，单击“Finish”按钮。

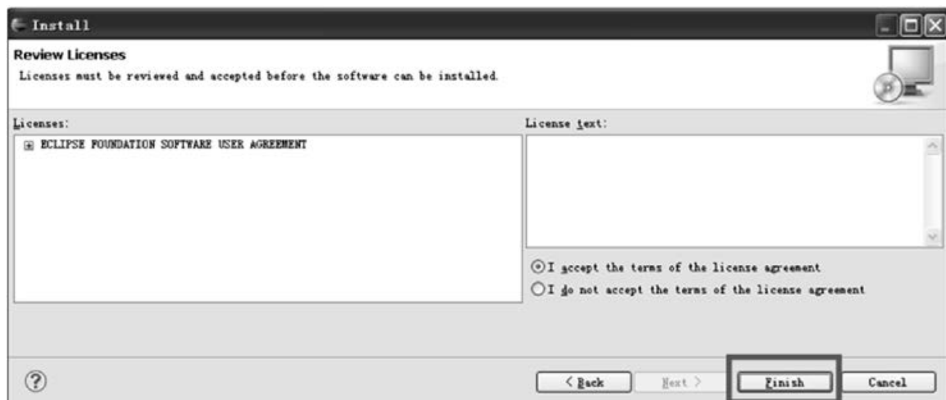


图 1-12　同意安装汉化包

4）重启 Eclipse 平台。安装需要一段时间，安装完成后会出现如图 1-13 所示的警告对话框，单击“OK”按钮，然后出现图 1-14 所示的重启平台对话框，单击“Restart Now”按钮重启 Eclipse 平台，汉化过程完成。

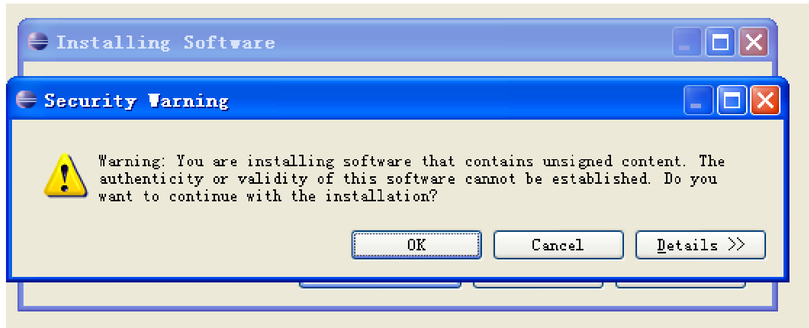


图 1-13　汉化警告对话框

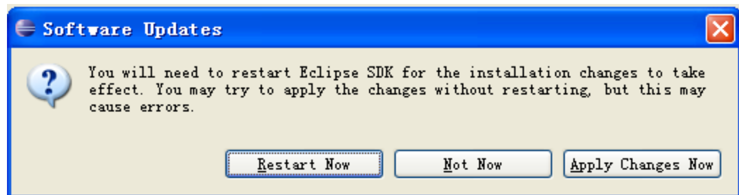


图 1-14　重启 Eclipse 平台对话框

（3）Android SDK 安装与配置

1）解压 Android SDK 到相应路径，如：D：\Program Files\android-sdk-windows，运行“SDK Manager.exe”。如果遇到消息为“Failed to fetch URL…”的错误提示，则需要将 HTTPS 方式改为 HTTP 方式。解决办法是在“Android SDK and AVD Manager”窗口的左侧选择“Settings”（如图 1-15 所示）选择“Force https：//…”，单击“Save & Apply”，并重启“SDK Manager.exe”。

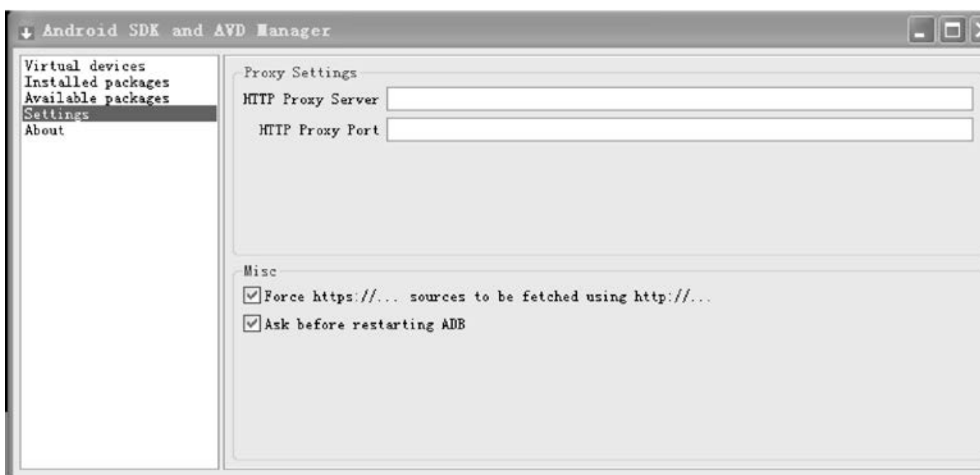


图 1-15　Failed to fetch URL 解决办法



2）单击“Android SDK and AVD Manager”左侧窗口中的“Availble Packages”，选择要安装的 API 版本、SDK 文档及 USB 驱动，为了便于全面测试，在这里选择全部安装，如图 1-16 所示。

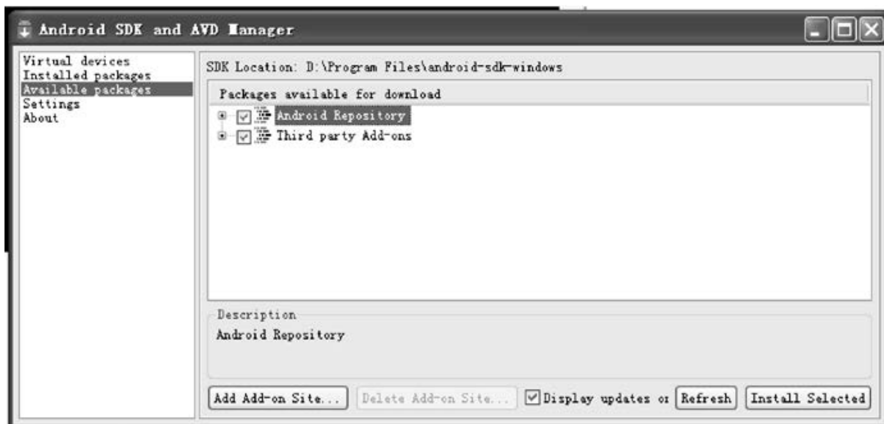


图 1-16　选择 API 版本

3）选择好之后，单击“Install Selected”按钮，在接下来的界面中依次单击“Accept All”和“Install Accepted”按钮，下载并安装 Android SDK。

4）配置 Android SDK 环境变量。首先，与配置 Java 环境变量一样：在“我的电脑”上单击鼠标右键→“属性”→“高级”→“环境变量(N)”（如图 1-3 所示）。其次，选择“系统变量”中的“path”项，单击“编辑”按钮，将 Android SDK 安装文件夹下的 tools 文件夹的路径添加到“path”中，前面要用“；”隔开，如图 1-17 所示，依次单击“确定”按钮完成配置。

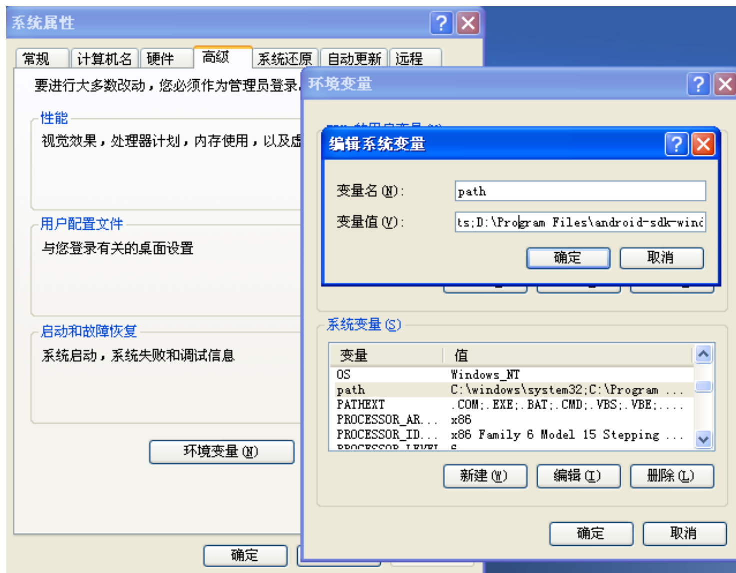


图 1-17　配置 Android SDK 环境变量

（4）安装和配置 ADT

安装 ADT 的方法和 Eclipse 汉化过程基本相同，具体步骤如下：

1）启动 Eclipse，因为已经将 Eclipse 汉化了，所以显示的是中文的样式。单击“帮助”菜单，依次选择“安装新软件（S）…”菜单命令，如图 1–18 所示。

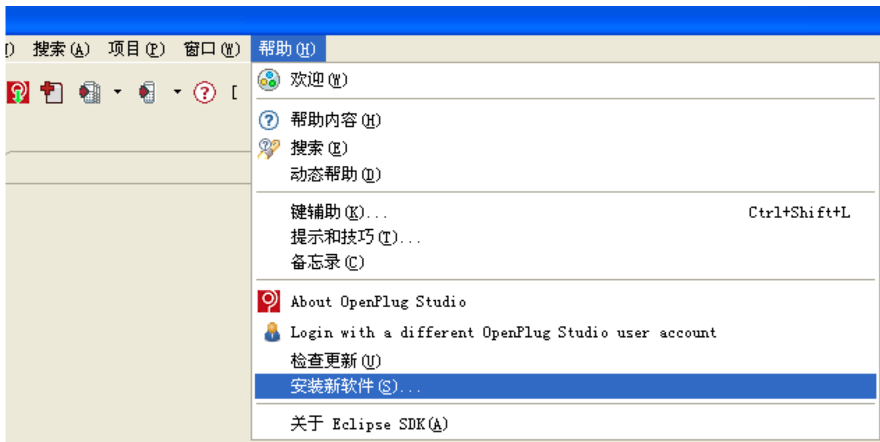


图 1–18　汉化后的“帮助”菜单下的“安装新软件（S）…”

2）打开对话框，选择“添加（A）…”项，弹出需要下载的 ADT 地址，如图 1–19 所示。给路径取一个别名，然后在“位置（L）”中输入地址，地址是：https：//dl–ssl.google.com/android/eclipse/，如图 1–20 所示。然后单击“确定”按钮，经过下载，在“名称”子项中会出现安装包，如图 1–21 所示。

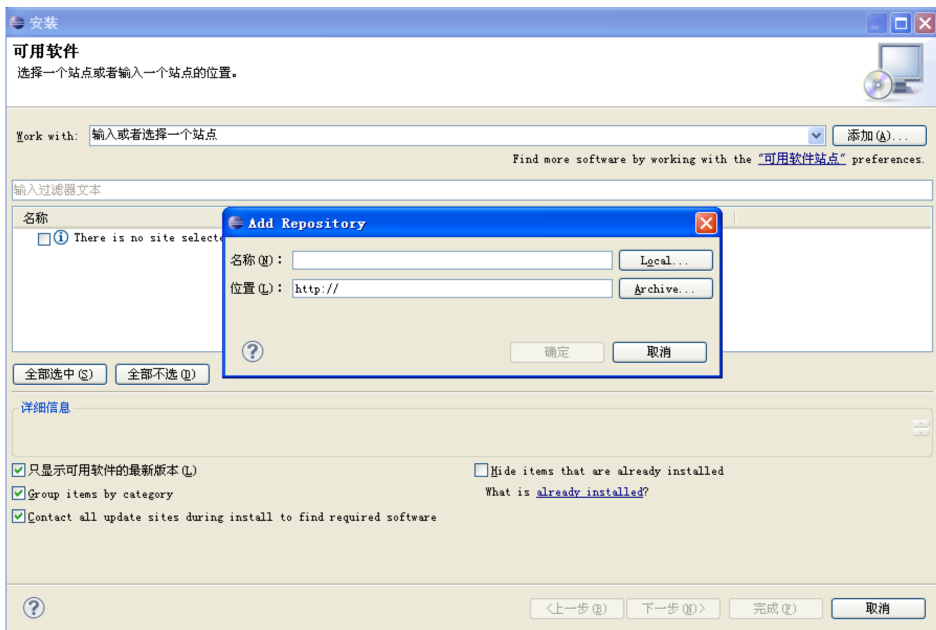


图 1–19　添加 ADT 包地址（1）



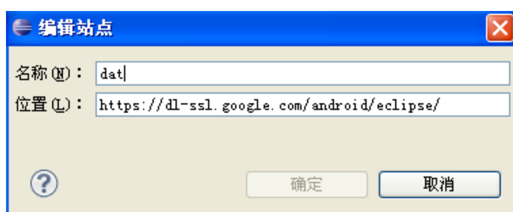


图 1-20　添加 ADT 包地址（2）

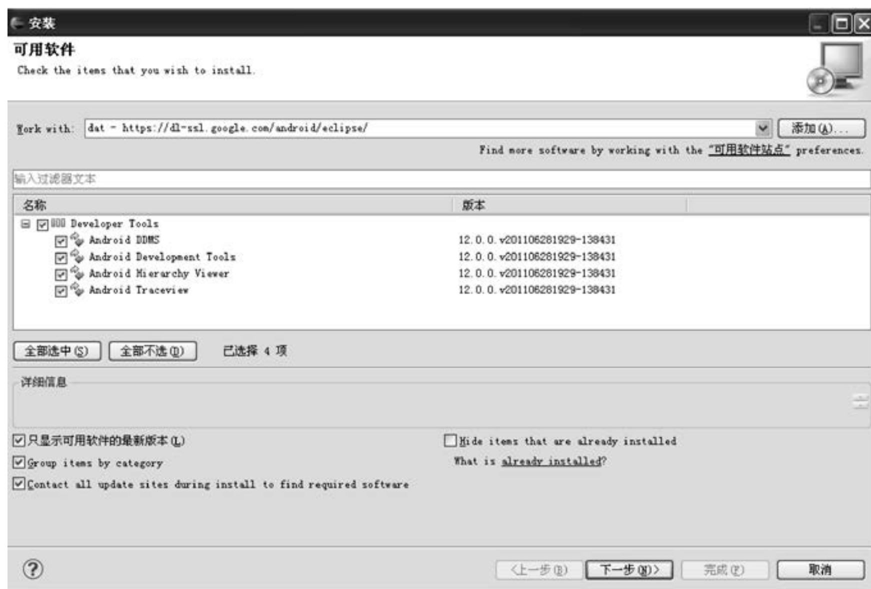


图 1-21　选择 ADT 安装包

3）单击“下一步”按钮，直到出现如图 1-22 所示的界面，选择“I accept the terms of the license agreement”，单击“Finish”按钮。同样会出现警告及重启 Eclipse 对话框，操作方法与汉化相同，这里不再赘述。

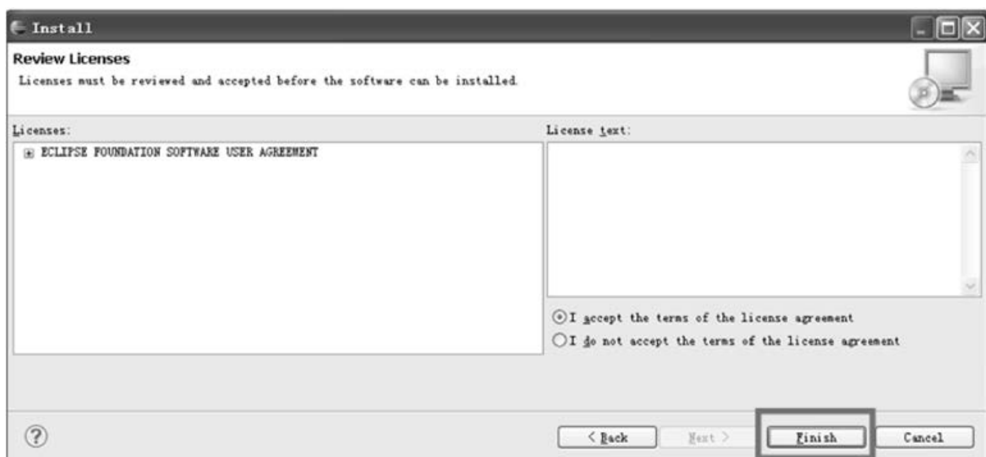


图 1-22　同意安装 ADT

4）配置 ADT。打开菜单“窗口”→“首参数”→“Android”，单击“浏览”按钮，选择 Android SDK 的安装路径，如图 1-23 所示。单击“确定”按钮即可。

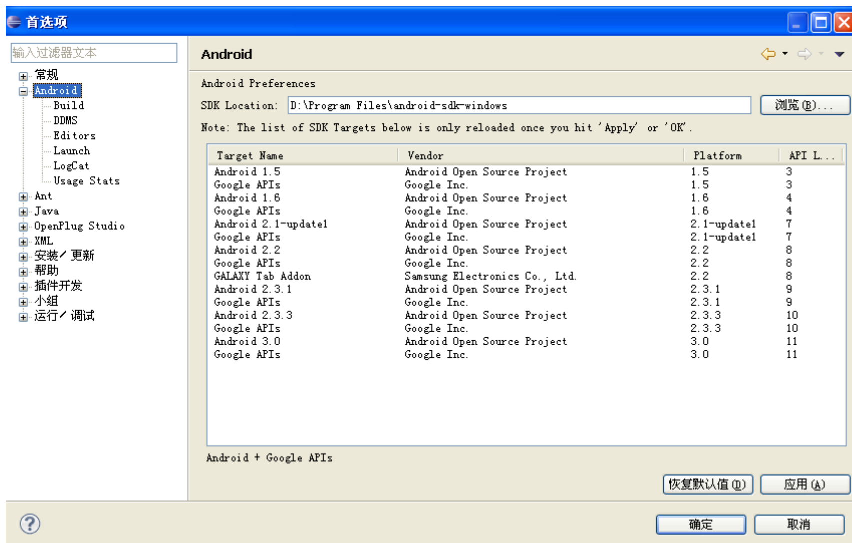


图 1-23　在 Eclipse 中通过安装好的 ADT 配置 Android SDK

5）验证配置是否成功。选择“文件”→“新建”→“项目”→“Android”→“Android Project”命令，则表示安装配置成功，如图 1-24 和图 1-25 所示。

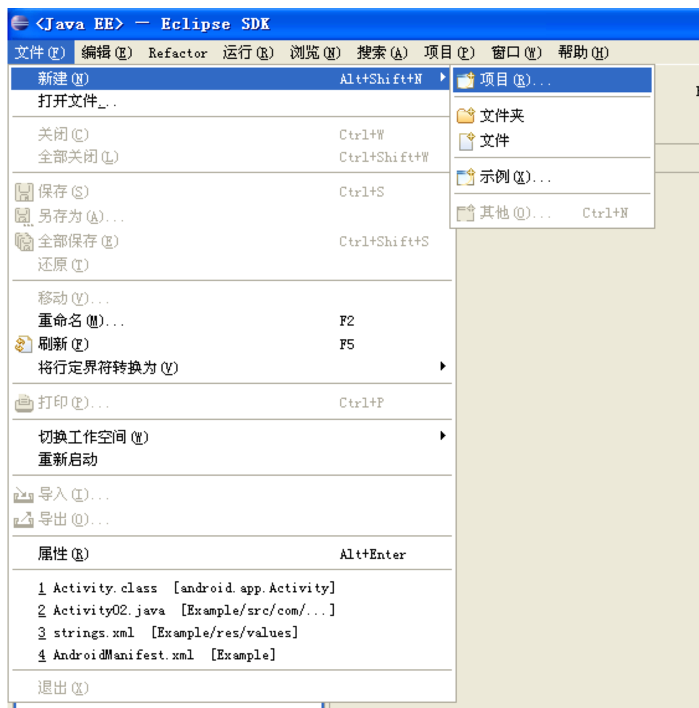


图 1-24　新建 Android 项目界面（1）



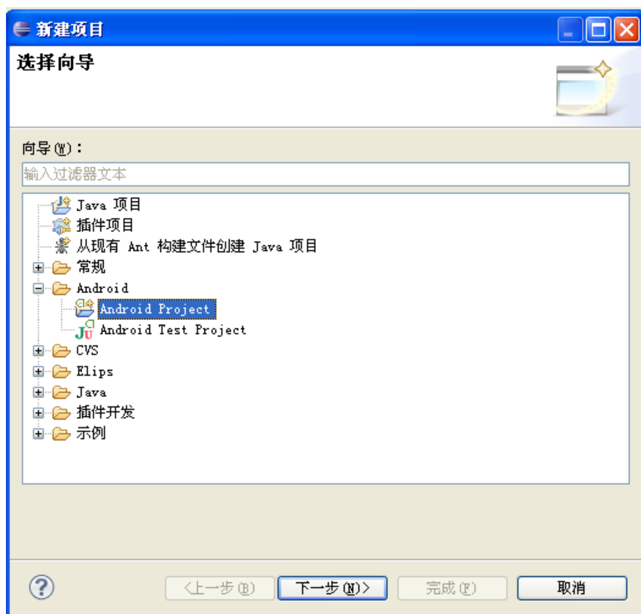


图 1-25　新建 Android 项目界面（2）

1.3　体验Android开发之旅

配置完成开发环境，相信大家都想一睹 Android 开发的神奇与风采吧！下面通过一个简单的示例让读者了解 Android 开发的整个过程。

1.3.1　创建HelloAndroid项目

前面讲述过 ADT 的作用，它将 Eclipse 和 Android SDK 连接在一起，起到桥梁纽带的作用，提供了生成 Android 应用框架的功能。我们现在使用 ADT 在 Eclipse 平台创建一个 Android 项目，步骤如下：

1　打开 Eclipse 开发工具，选择菜单项中的“文件”→“新建”→“项目”→“Android”→“Android Project”，如图 1-24、图 1-25 所示。

2　单击“下一步”按钮，在“Project name”文本框中输入“HelloAndroid”，然后再“Build Target”选项框中选择 Android SDK 的版本，这里选择“Android SDK 2.2”（可以选择您需要的其他 SDK 版本），在 properties 中的“Application name”文本框中输入应用程序的名字（HelloAndroid），在“Package name”文本框中输入应用程序包的名字（com.chapter1.helloandroid），在“Create Activity”文本框中输入 Activity 的名字（HelloAndroid），在“Min SDK Version”中填写 API Level（这个值要填正确，否则编译过后的.apk 文件不能在手机真机环境中正确运行，不同款式和类型的手机 Android 操作系统的版本不一样，该值就不一样），Android SDK 2.2 操作系统的值是 8，如图 1-26、图 1-27 所示。单击“下一步”按钮，完成创建，在 Eclipse 的“包资源管理器”中就会看到 Eclipse 自动生成的 HelloAndroid 项目。

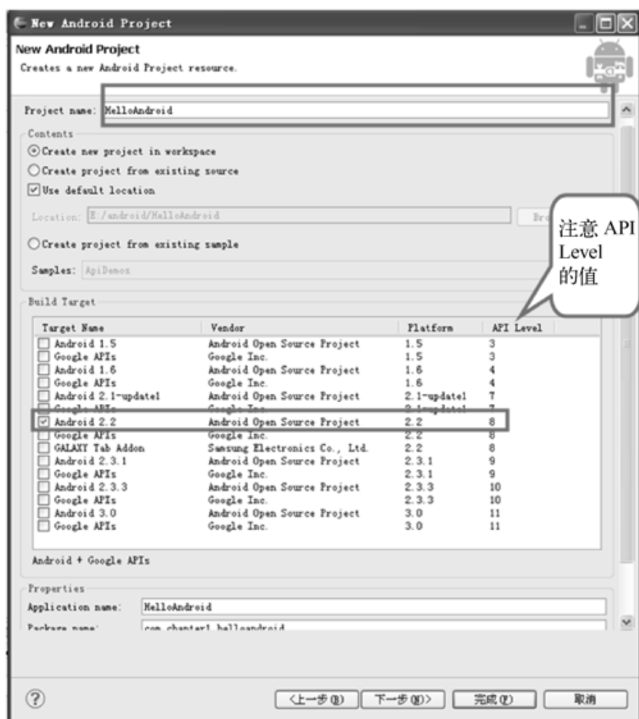


图 1-26　创建 HelloAndroid 项目（1）



图 1-27　创建 HelloAndroid 项目（2）



1.3.2　模拟器配置及运行HelloAndroid

创建好了一个简单的 HelloAndroid 项目，现在需要配置手机开发模拟器（AVD）。具体步骤如下：

1）单击菜单“窗口”→“Android SDK and AVD Manager”，如图 1-28 所示。

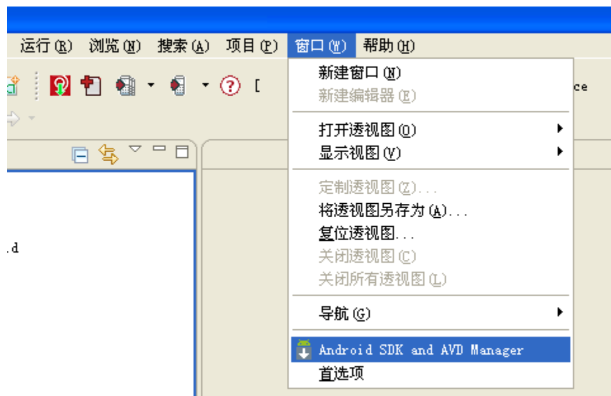


图 1-28　配置手机开发模拟器（AVD）

2）弹出图 1-29 所示配置对话框。单击左边的“Virtual devices”选项，再单击右边的“New…”按钮，新建一个手机模拟器（AVD）。

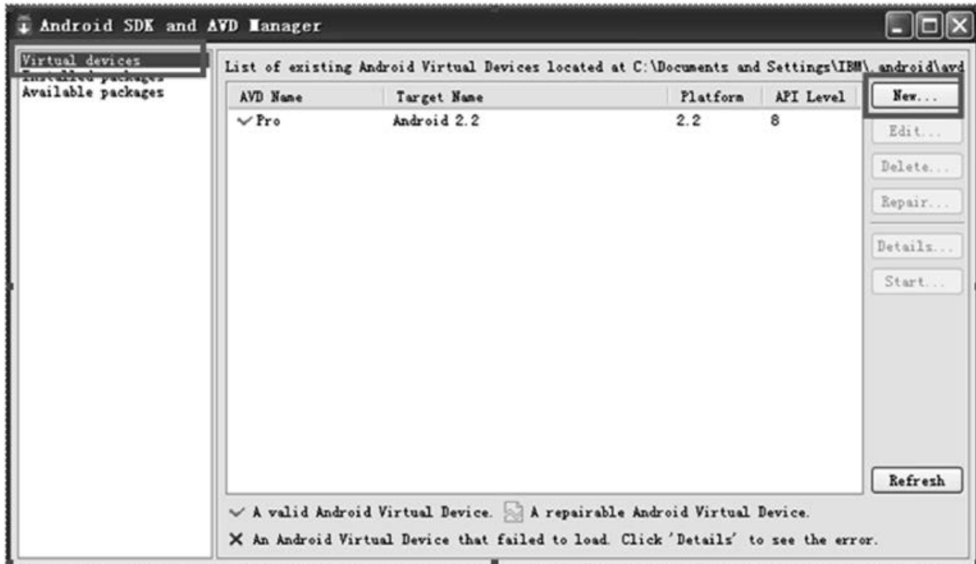


图 1-29　手机开发模拟器配置对话框

3）配置手机模拟器（AVD）。在弹出的“Create new Android Virtual Device（AVD）”中配置模拟器参数。“Name”中输入模拟器的别名，如“android”；在“Target”中选择模拟器的手机操作系统版本，在这里选择“Android 2.2 -API Level 8”；在“SD Card”中配置模拟器 SD 卡的大小；在“Skin”标签中设置模拟器的风格，如选择“HVGA”，然后单击

“Create AVD”，如图 1-30 所示。在创建的过程中请耐心等待，直到出现图 1-31 所示的对话框，则表示配置 AVD 成功。

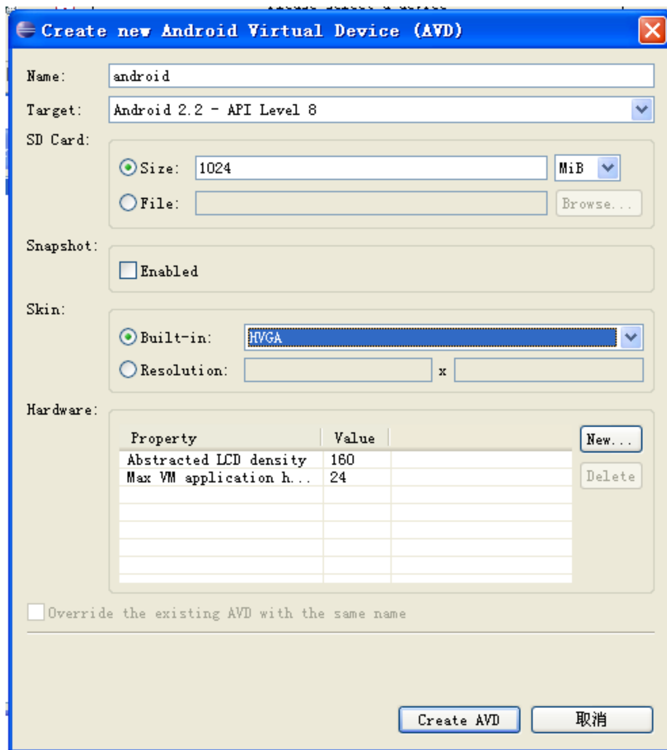


图 1-30　填写手机模拟器相关参数

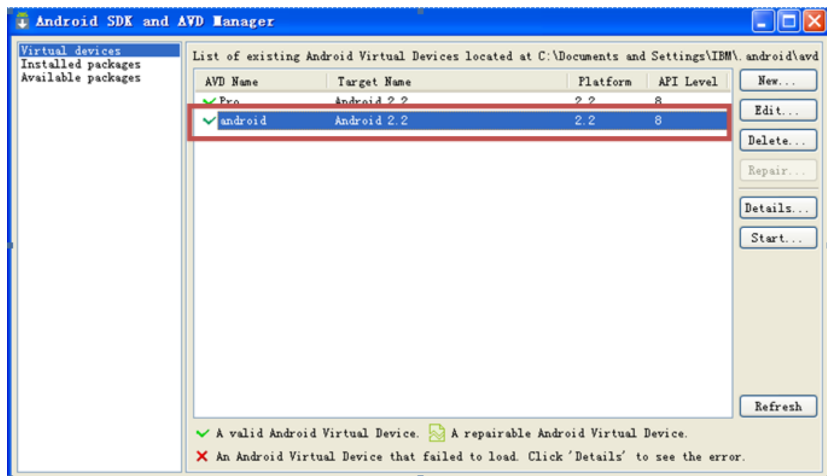


图 1-31　Android 手机模拟器配置成功

4）为 HelloAndroid 添加的模拟器及运行 HelloAndroid。具体操作是：用鼠标右键单击“HelloAndroid”项目→“运行方式”→“运行配置”（如图 1-32 所示），弹出配置对话



框。用鼠标右键单击左侧的“Android Application”，选择“新建”（如图 1-33 所示），弹出如图 1-34 所示的对话框。在“名称”中填写模拟器的名称，在 Project 中单击“Browse…”，选择 HelloAndroid 项目，如图 1-34 所示。到此完成 HelloAndroid 项目的模拟器配置。单击“运行方式”，启动 HelloAndroid 项目，运行成功第一个 Android 项目。加载模拟器需要一段时间，请耐心等待，直到模拟器上出现图 1-35 所示的文字——“Hello world，HelloAndroid！”，则表示第一个 Android 项目运行成功。

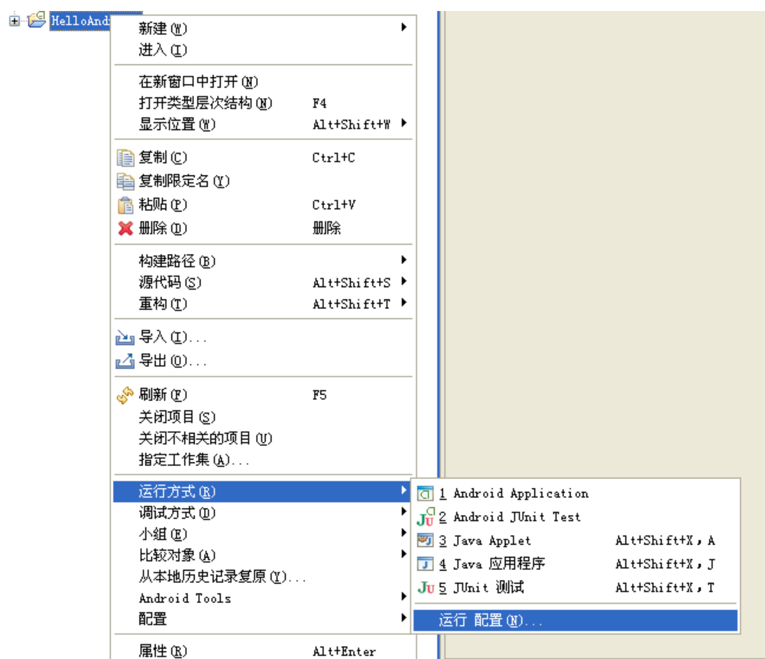


图 1-32　配置 HelloAndroid 模拟器（1）

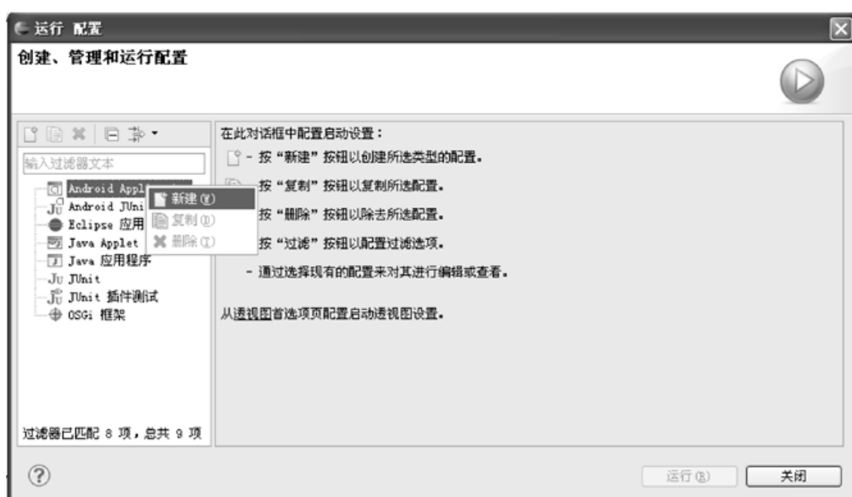


图 1-33　配置 HelloAndroid 模拟器（2）

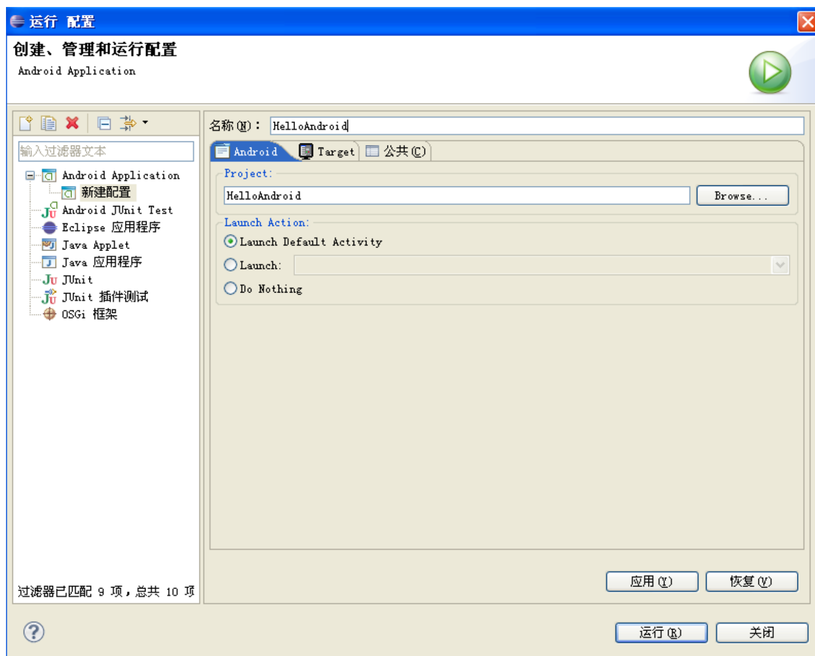


图 1-34　配置 HelloAndroid 模拟器（3）

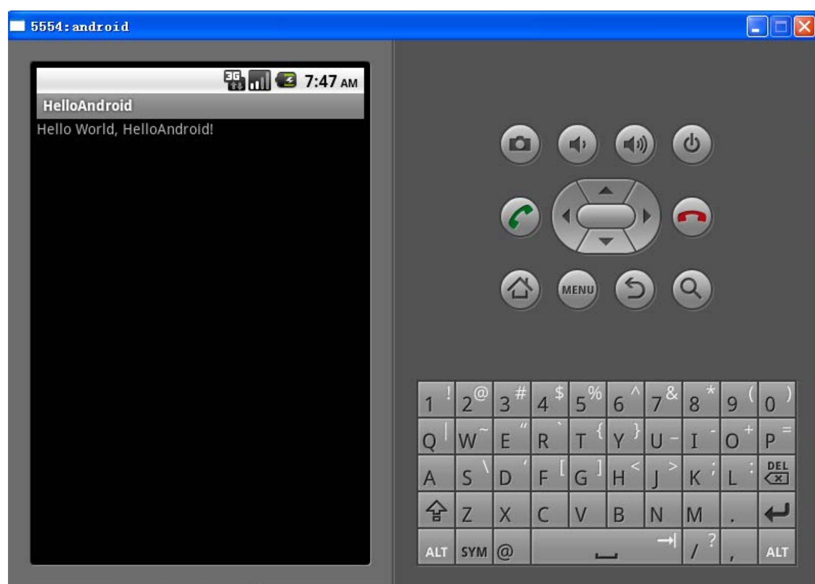


图 1-35　HelloAndroid 运行成功效果图

1.3.3　剖析HelloAndroid项目结构

看到第一个 Android 程序运行成功，相信读者也想赶快来开发属于自己的 Android 项目。但要彻底掌握开发规律，快速编写出属于自己的 Android 项目，首先要了解 Android 的项目结构。单击 HelloAndroid 项目左边的“+”，展开第一层项目结构，如图 1-36 所示。下



面详细讲解 HelloAndroid 项目下每一个文件夹的作用。

（1）src 文件夹

Android 项目遵循了现今普遍使用的三层架构模式（MVC 模式），在 src 文件夹下存放的就是“C（业务逻辑）”代码，它是以“.java”后缀名结尾的文件，如图 1-37 所示。打开 HelloAndroid.java 文件，可以看到相应的代码块。

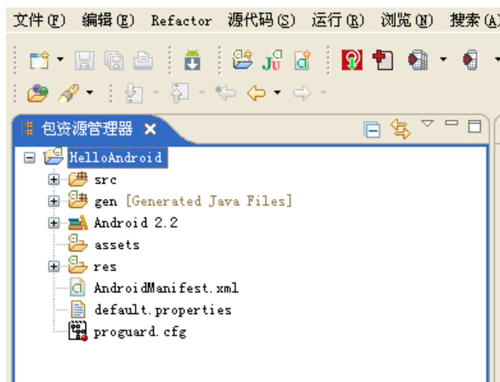


图 1-36　HelloAndroid 第一层结构

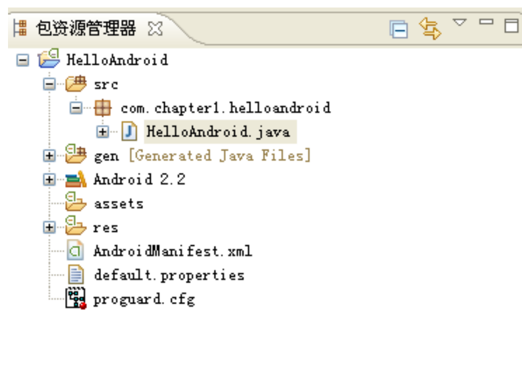


图 1-37　src 文件夹内容

下面我们重点分析 HelloAndroid 项目的主程序文件 HelloAndroid.java，如代码清单 1-1 所示。

代码清单 1-1　第 1 章HelloAndroid项目中的HelloAndroid.java文件

```
/*包名*/
package com.chapter1.helloandroid;
/*导入引用的包*/
import android.app.Activity;
import android.os.Bundle;
/**
 * HelloAndroid 程序的入口，继承 Activity
 * @author
 */
public class HelloAndroid extends Activity {
    /*  重写 Activity 中的 onCreate 方法  */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*  设置 Activity 要显示的布局为 R.layout.main，该布局在 res 下的 layout 中创建*/
        setContentView(R.layout.main);
    }
}
```

在代码清单 1-1 中，主程序 HelloAndroid 类继承了 Activity 类，重写了 void onCreate (Bundle savedInstanceState)方法。在 onCreate 方法中通过设置 setContentView(R.layout.main) 来显示画面，R.layout.main 表示界面布局，该布局在 res 下的 layout 中创建（res\layout\main.xml）。

（2）gen 文件夹

gen 文件夹如图 1-38 所示。存放的是系统自动生成的项目索引文件，如在 res\layout\main.xml 中定义一个文本框（TextView）id 为“@+id/TextView1”则会在 R.java 中生成对应的代码：“public static final int TextView1=0x7f050000;”，具体代码如代码清单 1-2 所示。

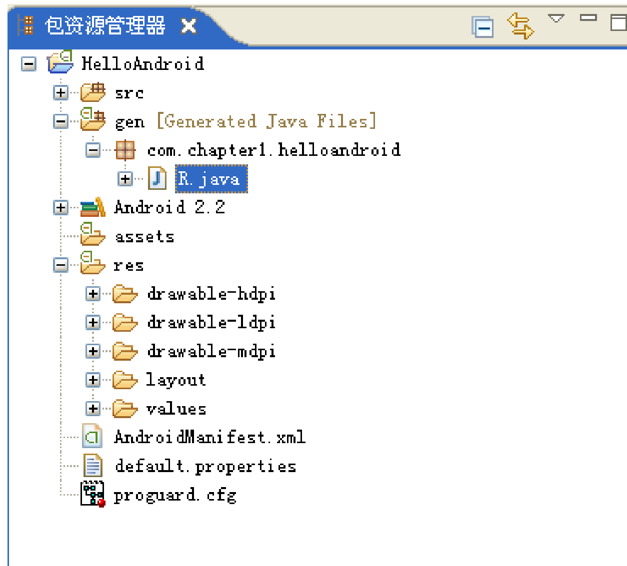


图 1-38　gen 文件夹内容

代码清单 1-2　第 1 章HelloAndroid项目中的R.java文件内容

```
package com.chapter1.helloandroid;

public final class R {
    public static final class attr {
    }
    public static final class drawable {
        public static final int icon=0x7f020000;
    }
    public static final class id {
        public static final int TextView1=0x7f050000;
    }
    public static final class layout {
        public static final int main=0x7f030000;
    }
    public static final class string {
        public static final int app_name=0x7f040001;
        public static final int hello=0x7f040000;
    }
}
```

需要注意的是，R.java 是自动生成的索引文件，这个文件是只读模式，不能更改，它是 Android 项目文件夹下各个对象的索引。如：public static final class drawable 方法下存在的是



res/drawable 文件的索引，public static final class id 方法下存在的是所有取了 id 名称的文件的索引。

（3）Android 2.2 包

之所以在 HelloAndroid 项目中会生成 Android 2.2 包资源文件，那是因为之前选择的是 Android 2.2 API，该包下存放的是与项目开发有关的底层包。可以说所有 Android 应用程序开发都是基于这个包进行的，我们可以调用包里面的方法与属性，实现更多的功能，如图 1-39 所示。

（4）assets 文件夹

assets 文件夹存放的是不进行编译加工的原生文件，即表示该文件夹里面的文件不会像 xml 文件一样被预编译，主要存放的是一些图片及 html、js、css 等文件。在后面的章节中会介绍如何读取 assets 文件夹下的资源。

（5）res 文件夹

res 文件夹下存放的也是资源文件，这些资源文件与 assets 文件夹下存放的资源文件的不同之处在于：res 下存放的是需要进行编译加工的原生文件，而 assets 下存放的是不需要进行编译加工的原生文件。在 res 文件夹下默认有 drawable-hdpi、drawable-mdpi、drawable-ldpi、layout 及 values 等文件夹，如图 1-40 所示。下面将一一介绍各个部分的功能。

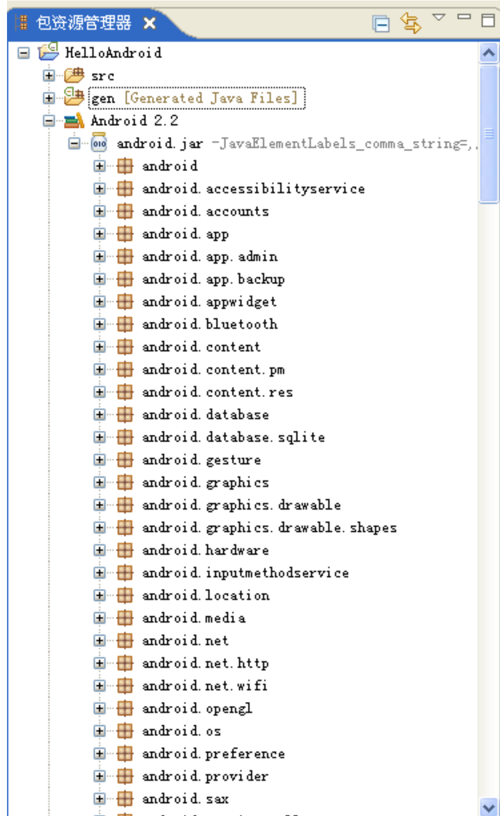


图 1-39 Android 2.2 包内容

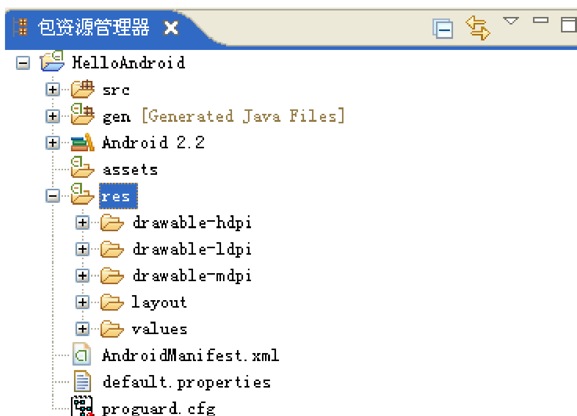


图 1-40 res 文件夹内容

drawable-hdpi、drawable-mdpi、drawable-ldpi 一般用来存放图片，Android 操作系统的手机可能采用不同的手机分辨率，因此在开发过程中会要求使用具有不同分辨率的图片。

drawable-hdpi 中存储的是高分辨率的图片，drawable-mdpi 中存储的是中等分辨率的图片，而 drawable-ldpi 中存储的是低分辨率的图片，所以，开发者需要把相同的图片用图片处理软件处理成不同分辨率的图片，然后保存在对应的文件夹中。当然，如果属于学习阶段，也可以在三个目录下保存相同的图片。

在 layout 文件夹下存放的是外观文件，相当于三层架构模式（MVC）中的“V 层代码。刚才我们所说的 main.xml 文件就是放在该层下，如代码清单 1−3 所示。从代码清单中可以看出：该文件是一个 xml 文件，在程序的开头需要加入<?xml version="1.0" encoding="utf-8"?>xml 文件标志，接下来的代码片段才是 xml 布局的正文代码，主要定义了相应的控件及布局，这一部分的代码内容将在接下来的章节中使用与讲解。

代码清单 1−3　第 1 章HelloAndroid项目中的main.xml文件

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
<TextView
     android:id="@+id/TextView1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"
     />
</LinearLayout>
```

在 values 文件夹下自动生成一个“strings.xml”文件，在该文件中常常定义一些常量，如程序的标题等，如代码清单 1−4 所示。看到 resources 节点下的两行代码是不是感觉似曾见过？这就是模拟器上显示的文字，“Hello World， HelloAndroid!”。需要注意的是，所有的常量都必须写在 resources 节点下。

代码清单 1−4　第 1 章HelloAndroid项目中的strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
     <string name="hello">Hello World, HelloAndroid!</string>
     <string name="app_name">HelloAndroid</string>
</resources>
```

（6）AndroidManifest.xml

AndroidManifest.xml 配置文件，它是每个 Android 程序中必需的文件，如代码清单 1−5 所示。它位于 application 的根目录下，描述了 package 中的全局数据，除了包括该项目中常使用的 activity、services、receiver，providers 及 intent receivers，还可以指定 permissions、instrumentation 等成员。表 1−3 和表 1−4 列出了所有成员的具体含义功能。

代码清单 1−5　第 1 章HelloAndroid项目中的AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chapter1.helloandroid"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".HelloAndroid"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
</manifest>
```

AndroidManifest.xml 是一个 xml 文件，因此在文件的第一行包含了 xml 文件的声明。第一个节点<manifest>，几乎所有的 AndroidManifest.xml（以及许多其他 Android 的 xml 的文件）在第一个元素中包含了命名空间的声明 xmlns：android=http：//schemas.android.com/apk/res/android，这样使得 Android 中各种标准属性能在文件中使用，提供了大部分元素中的数据。

在<manifest>节点下默认包含<application>和<uses-sdk>节点，可以声明其他节点，以实现更多的功能。大部分<manifest>包含了单个<application>的元素，它定义了所有的 application 级别组件和属性，并能在 package 中使用。<application>节点下一个很重要的功能就是它包含<activity>节点，任何被用户看做顶层应用程序，并能被程序启动器所用的 package，需要包含至少一个 Activity 组件来支持 MAIN 操作和 LAUNCHER 种类，且在该节点下定义了那些在 src 中定义的 java 文件。如在 src 中使用到了一个 HelloAndroid.java，它继承了 Activity，并设置了它要显示的布局是 layout 下的 main.xml，那就必须在<application>节点下定义一个<activity>，其中 android:name=".HelloAndroid"，就是指 HelloAndroid.java 的名称。因此，若要在界面上运行继承了 Activity 的文件对象，都必须在配置文件中声明一个<activity>节点，如果没有，则应在 AndroidManifest.xml 中声明一个<activity>节点，否则该 activity 文件将不能运行。

在<activity>节点下使用<intent-filter>，就是让应用程序组件告诉 Android 底层系统，它们能为其他程序组件的动作请求提供服务，包括同一个程序的组件、本地的或第三方的应用程序。为了注册一个应用程序组件为 Intent 处理者，在组件的 manifest 节点添加一个<intent-filter> 标签。在该节点里使用下面的标签（关联属性）就能指定组件支持的动作、种类和数据。可以说，一条<intent-filter>元素至少应该包含一个<action>，否则任何 Intent 请求都不能和该<intent-filter>匹配。如果 Intent 请求的 Action 和<intent-filter>中的某一条<action>匹配，那么该 Intent 就通过了这条<intent-filter>的动作测试。<intent-filter>节点下还包含<category>，指定应用程序默认启动的 Activity。<uses-sdk>则定义了应用程序所使用的 SDK 版本，也就是前面讲的 API Level 的值。AndroidManifest.xml 配置文件的默认详细功能如表 1-3、表 1-4 所示。

表 1-3　AndroidManifest.xml 默认配置文件详情

项	功能说明
<manifest>	根节点，描述了 package 中所有的内容
xmlns:android	包含命名空间的声明。代码如：xmlns:android=http：//schemas.android.com/apk/res/android，功能是使 Android 中各种标准属性能在文件中使用，提供了大部分元素中的数据
package	src 下定义的应用程序包
versionCode	定义版本代码
versionName	定义版本名称
<application>	级别组件声明的根节点。此元素可包含在 application 中定义的全局和默认的属性，如标签，icon，主题，必要的权限等。一个 manifest 能包含零个或一个此元素（不允许多于一个）
android:icon	应用程序图标
android:label	应用程序名字
<activity>	是用来与用户交互的主要工具。当用户打开一个应用程序的初始页面时，就要继承 activity。大部分被使用到的其他页面也由不同的 activity 所实现并声明在另外的 activity 标记中。要注意的是：每一个 activity.java 程序文件必须要与一个<activity>标记对应，无论它给外部使用或是只用于自己的 package 中。如果没有对应的标记，程序将不能运行。另外，为了支持运行时查找 activity，可以使用一个或多个<intent-filter>元素来描述 activity 所支持的操作
<intent-filter>	声明了指定的一组组件所支持的 Intent 值，从而形成了 IntentFilter。除了能在此元素下指定不同类型的值，属性也能放在此处来描述一个操作所需的唯一的标签，如 icon 或其他信息
<action>	组件支持的动作
<category>	指定应用程序默认启动的 activity
<uses-sdk>	定义了应用程序所使用的 SDK 版本，也就是前面所讲的 API Level 的值

表 1-4　AndroidManifest.xml 其他可配置的选项

可选项	功能说明
uses-permission	请求程序 package 正常运行时所需赋予的安全许可，使用 SecurityModel 来获得许可的更多信息。一个 manifest 能包含零个或更多此元素
permission	声明了安全许可来限制哪些程序在 package 中的组件和功能，使用 SecurityModel 来获得许可的更多信息。一个 manifest 能包含零个或更多此元素
instrumentation	声明了用来测试此 package 或其他 package 指令组件的代码，使用 Instrumentation 来获得许可的更多信息。一个 manifest 能包含零个或更多此元素
receiver	它能使 application 获得数据的改变或者发生的操作，即使当前应用程序不在运行状态。利用 activity 标记，可以选择包含一个或多个 receiver 所支持的<intent-filter>元素
service	service 能在后台运行任意时间的组件。利用 activity 标记，可以选择包含一个或多个 receiver 所支持的<intent-filter>元素
provider	contentProvider 是用来管理持久化数据并发布给其他应用程序使用的组件

（7）default.properties

default.properties 文件记录项目中所需要的环境信息，比如 Android 的版本类型等。一般 default.properties 文件代码中的注释已经把 default.properties 解释得很清楚。如果某个项目没有 default.properties 这个文件，可以直接从其他项目中复制过来使用。“target=android-8”表示使用 android 2.2 平台开发。下面为 HelloAndroid 项目中 default.properties 的代码清单 1-6

代码清单 1-6　HelloAndroid项目中default.properties

```
# This file is automatically generated by Android Tools.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file must be checked in Version Control Systems.
```



```
#
# To customize properties used by the Ant build system use,
# "build.properties", and override values to adapt the script to your
# project structure.
# Project target.
target=android-8
```

（8）proguard.cfg 文件

在讲述 proguard.cfg 文件内容之前，先介绍 proguard.cfg 的功能，它具有实现 Android Java 混淆器的功能。Java 的字节码一般是非常容易反编译的，为了保护 Java 源代码，往往会对编译好的 class 文件进行混淆处理。ProGuard 的主要作用就是混淆，它是一个 Source Forge 上非常知名的开源项目，网站地址是：http：//proguard.sourceforge.net/。当然它还有对字节码进行缩减体积、优化等功能。

在 Android 2.3 以前， Android 代码只能手动添加 ProGuard 来实现代码混淆，非常不方便。而 Android 2.3 以后，Google 已经将这个工具加入到了 SDK 的工具集里。具体路径：SDK\tools\proguard。当创建一个新的 Android 工程时，在工程目录的根路径下，会出现一个 ProGuard 的配置文件 proguard.cfg。也就是说，我们可以通过简单的配置，在 eclipse 工程中直接使用 ProGuard 混淆 Android 工程。具体混淆的步骤也非常简单，只需要在工程描述文件 default.properties 中添加一句话，启用 ProGuard，如代码清单 1-7 所示。

代码清单 1-7　在HelloAndroid项目的default.properties中增加混淆功能

```
# This file is automatically generated by Android Tools.
# Do not modify this file -- YOUR CHANGES WILL BE ERASED!
#
# This file must be checked in Version Control Systems.
#
# To customize properties used by the Ant build system use,
# "build.properties", and override values to adapt the script to your
# project structure.
# Project target.
target=android-8
proguard.config=proguard.cfg
```

这样，ProGuard 就可以使用了。当我们正常通过 Android Tools 导出 Application Package 时，ProGuard 就会自动启用，从而达到混淆所编写的代码的功能。

关于 proguard.cfg 配置，进一步思考混淆后的结果就会发现：如果那些提供给外部的类、方法、变量等名字被改变，那么程序本身的功能就无法正常实现。而 ProGuard 如何知道哪些是可以改名，哪些是不能改变的呢？这主要是靠 proguard.cfg 文件来进行配置。在 Android 工程中，默认自动生成的 proguard.cfg 已经针对 Android 的一般情况进行了配置，下面我们打开这个配置文件，具体内容如代码清单 1-8 所示。

代码清单 1-8　HelloAndroid项目中proguard.cfg配置文件的内容

```
-optimizationpasses 5
-dontusemixedcaseclassnames
-dontskipnonpubliclibraryclasses
```

```
-dontpreverify
-verbose
-optimizations !code/simplification/arithmetic,!field/*,!class/merging/*

-keep public class * extends android.app.Activity
-keep public class * extends android.app.Application
-keep public class * extends android.app.Service
-keep public class * extends android.content.BroadcastReceiver
-keep public class * extends android.content.ContentProvider
-keep public class * extends android.app.backup.BackupAgentHelper
-keep public class * extends android.preference.Preference
-keep public class com.android.vending.licensing.ILicensingService

-keepclasseswithmembernames class * {
    native <methods>;
}

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet);
}

-keepclasseswithmembernames class * {
    public <init>(android.content.Context, android.util.AttributeSet, int);
}

-keepclassmembers enum * {
    public static **[] values();
    public static ** valueOf(java.lang.String);
}

-keep class * implements android.os.Parcelable {
  public static final android.os.Parcelable$Creator *;
}
```

它主要保留了继承自 Activity、Application、Service、BroadcastReceiver、ContentProvider、BackupAgentHelper、Preference 和 ILicensingService 的子类。之所以保留这些子类，是因为这些子类都可能被外部调用。另外，它还保留了含有 native 方法的类、构造函数从 xml 构造的类（一般为 View 的子类）、枚举类型中的 values 和 valueOf 静态方法、继承 Parcelable 的跨进程数据类。在实际的工程项目中，可能 Google 自动生成的配置不能胜任我们的混淆工作。所以，往往需要自己编写一些 ProGuard 配置，这方面更详细的资料可以在 Google 官网上的 Manual→Usage 里找到。

1.4 设置Android模拟器中文环境

在开发过程中，我们希望在开发环境中使用的模拟器与真实手机所使用的开发界面一样，这样做有助于解决中文乱码的问题，而英文是不会乱码的。从刚才所运行的第一个 Android 程序（HelloAndroid）的界面中我们会发现，模拟器显示的全是英文界面，如何设置 Android 模拟器的中文环境呢？ 本书使用的是 Android 2.2 模拟器，下面以 Android 2.2 为例



讲解如何设置模拟器的中文环境。

1）单击模拟器上“返回”键回到主屏幕，再单击“Menu”键，打开菜单，如图 1-41 所示。

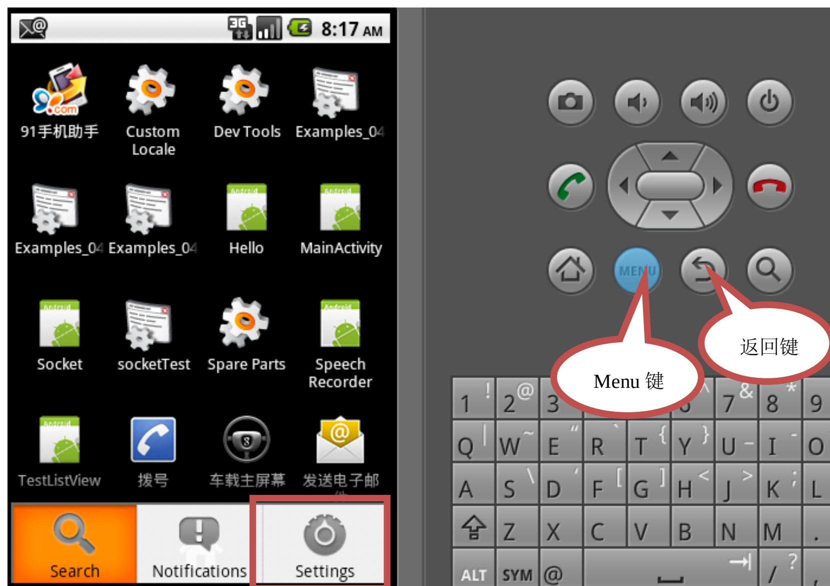


图 1-41　Settings 位置界面

2）选择“settings”，再选择“Language & Keyboard”项，如图 1-42 所示。

3）选择“Select language”，如图 1-43 所示，找到“中文（简体）”，这样就将模拟器设置成中文环境，如图 1-44 所示。

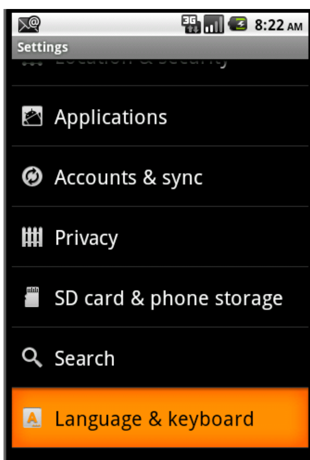


图 1-42　Language & Keyboard 位置界面



图 1-43　Select language 位置界面



图 1-44　选择中文（简体）

1.5　关于本书

下面介绍本书的主要内容。

第 1 章介绍了 Android 开发的行业发展前景及就业前景，手把手地教会读者如何搭建与配置 Android 开发平台，接着又详细地剖析了第一个 Android 程序，最后介绍如何设置 Android 模拟器中文环境，让读者尽快上手进行 Android 开发。

第 2 章介绍了案例项目的开发流程，展示了案例项目的运行效果，让读者了解本书案例项目所涉及的整体技术和功能，本章拉开了“案例导向式”的 Android 项目开发的序幕。按项目功能划分进行层层分解，每一个功能界面都涉及一定量的基础控件。本章还介绍了通信后台使用的技术——ASP 技术，对 ASP 通信原理进行概括性的描述。

第 3 章开始进入 Android 开发实战阶段，即更加注重功能实现。介绍了项目主界面实现方法。通过在基础控件里讲解 GridView 的使用，在重点剖析中讲解数据适配器、事件处理机制以及页面切换的具体操作，让读者轻松完成项目主界面的设计。

第 4 章介绍了登录功能的实现。在基础控件中重点讲解文本框（TextView）、文本编辑控件（EditText）、单选按钮（RadioButton）及按钮控件（Button）；在重点剖析中讲解了 Android 长度单位、Android 布局，Android 数据存储以及 Android 如何与后台进行通信。这一章是本书的重点，涉及的知识点多。在这一章的最后一节也给出了如何组装数据与 ASP 后台进行通信以及 ASP 后台的实现过程。

第 5 章介绍了注册功能的实现。在基础控件中讲解 Activity 页面传值和信息如何使用正则表达式进行验证，让读者轻松地完成注册功能。本章起到承上的作用，进一步学习与巩固第 3 章所讲知识。

第 6 章介绍了选择功能的实现，通过在基础控件中讲解的自动匹配（AutoCompleteTextView）和列表控件（ListView）的使用，在重点剖析中进一步讲解了数据适配器中 ListAdapter 和 SimpleCursorAdapter 的使用，让读者轻松地完成城市选择功能。这一章也是本书的重点章节。

第 7 章介绍了查询功能的实现，通过在基础控件中讲解图片视图（ImageView）、图片按钮（ImageButton）、下拉列表（Spinner）、日期和时间控件（DatePicker、TimePicker）、菜单（Menu）、对话框（Dialog）以及进度条（ProgressBar、RatingBar），在重点剖析中讲解线程（Thread）handler 和 Runnable 的使用，让读者学会机票查询功能的实现方法。这一章涉及的知识点也很多，也是本书的重点及难点部分。

第 8 章介绍了信息列表功能的实现，通过在基础控件中讲解日期格式处理与转换、带图片多行 ListView 子项的使用，完成信息列表功能。

第 9 章介绍了机票 List 详情功能的实现，通过在基础控件中讲解文字交换器（TextSwitcher）和切换图片（ImageSwitcher）的使用方法，制作文字滚动和图片滚动效果，让读者体会 Android 组件功能的精彩动画效果及功能的完善。

第 10 章介绍了用户信息选择与填写功能的实现，通过在基础控件中讲解多选按钮（CheckBox）以及卷轴视图-动态添加子项（ScrollView），在重点剖析中讲解如何动态增加控件以及用户身份证的验证，从而完成用户信息选择与填写功能。

第 11 章介绍了案例项目未涉及的几个控件的用法，分别是状态栏提示（Notification、NotificationManager）、拖动条（SeekBar）以及拖动图片列表效果（Gallery），让读者体会制作滚动相册、消息提示等小程序的实现过程。

第 12 章介绍了如何美化界面，如何进行自适应处理，如何设置程序 Logo 以及如何进行



Android 测试，进一步完善项目。

第 13 章介绍了 Android 专题开发，专题开发包括那些在项目中未涉及，但又相对重要的特色开发，如 Android 数据存储，多媒体播放器的制作及 Android 外部接口编程等。

附录 A 讲述了如何在 Windows XP 和 Windows 7 中部署 IIS 以及如何发布网站等 Mobile/Server 应用项目中必须了解的经典技术。

附录 B 介绍一款较好的手机程序安装辅助工具——91 手机助手，使用它可以将项目安装到实际手机里并且将手机的运行结果显示在计算机屏幕或投影仪上，以展示手机项目开发的成果。

附录 C 介绍了 DDMS（Dalvik Debug Monitor Service）的常用功能。可以为测试设备截屏，针对特定的进程查看正在运行的线程以及堆信息、Logcat、广播状态信息、模拟电话呼叫、接收 SMS、虚拟地理坐标等功能。

第 2 章　Android项目开发——以手机订票系统为例

从这一章开始，将学习手机订票系统。该项目在编写过程中，就注意要做到全面、具体覆盖 Android 基础控件的使用。之所以这样编排，有过 Web 项目开发经验的人，相信都会有这样的感受，Web 项目开发存在着某些模板，有几个共同的模块是通常都有的，比如登录、注册模块及具体功能模块。Android 应用程序项目开发也是如此，存在着某一规律可供学习。本书的特色是站在项目开发的角度学习 Android 手机开发，弱化了 Android 基础控件部分，强调了 Android 应用程序项目开发的整体性，系统性地介绍项目开发的过程。本书以手机订票系统为主线，分功能模块来介绍 Android 项目开发，每一个功能模块又详细地讲解界面所涉及的控件的使用及实现该功能模块需要掌握的重点，在准备配料的基础上，完成项目开发。使用一个具体项目来讲述 Android 项目开发，其目的是抛砖引玉，让更多的人学会 Android 项目开发的流程，以便开发属于自己的具体项目。

2.1　项目需求概述

本书以一个具体的案例为主线，贯穿于 Android 应用程序开发过程中，在进行项目开发之前，首先要进行项目需求分析。在实际项目开发文档中，项目需求分析是必不可少的，而且相对比较全面，我们的目的是为了介绍 Android 项目开发的过程，因此只做简单的项目需求分析。

随着人们生活水平的提高，出差、旅行更多的是选择乘坐飞机这一种交通工具。它能够节省人们的行程时间，使人们的行程显得轻松。由于航空系统是一个全球的系统，其票务量之大，利润之大已经形成了一个行业。随着移动商务的发展，手机订票的社会需求也越来越强烈。因此，有必要开发一个手机订票系统，方便人们的出行。

2.2　项目功能分解

本书是一个模仿其他订票系统所开发的一款学习型项目。该项目的功能流程图如图 2-1 所示，包括以下几大功能模块：程序主界面功能、登录功能、注册功能、选择功能、查询功能、信息列表功能、List 详情（机票详情）界面、用户信息选择与填写功能、预订成功功能等，在进行项目开发之前，先具体阐述每一个界面要实现的效果图。

2.2.1　程序主界面功能

相比于计算机屏幕，手机屏幕确实比较小，一个手机应用型软件很少能在一个界面提供像门户网站那么大的信息量。因此，手机应用程序（如手机 QQ、携程订票）都习惯把所有功能罗列在一个主界面上，这样做的目的是使用户能够直观明了地知道该系统拥有哪些功能模



块。本项目使用的测试手机是 Samsung I5700，图 2-2 是手机订票系统主界面实现的结果图。

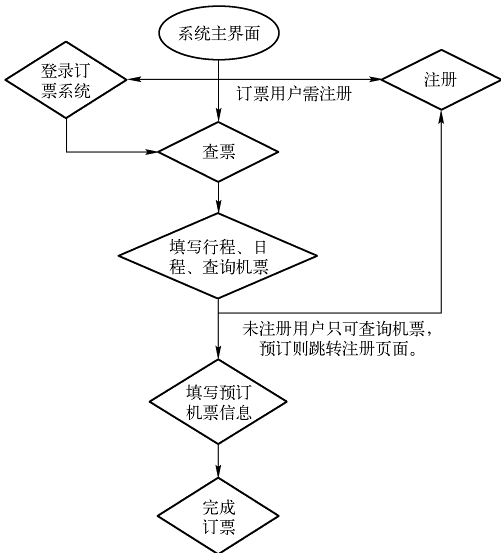


图 2-1　手机订票系统操作流程图



图 2-2　手机订票系统主界面

2.2.2　登录功能

大部分的网站都有登录界面，在订票系统主界面中选择“注册登录”按钮，进入注册登录界面。该界面实现用户登录的功能，从而提取注册用户的更多信息，如电话号码等信息。这样做方便用户在购买商品时，方便送货上门时联系购买方，因此有必要学习如何实现登录功能，界面实现效果如图 2-3 所示。

2.2.3　注册功能

如果是已注册用户，即可凭用户名、密码直接登录。如果是非注册用户，双击屏幕右下方“免费注册”按钮，进行用户注册界面。登录与注册是绑定在一起的，一般先注册再登录，注册的目的是区分普通用户和使用该系统用户，这也是购买性网站必不可少的部分。注册功能，界面实现效果如图 2-4 所示。



图 2-3　登录功能界面

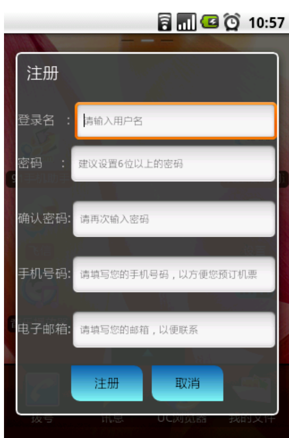


图 2-4　注册功能界面

2.2.4　选择功能

选择界面是依附于机票查询界面，它是机票查询界面的一项子功能，之所以单独拿出来作为一个功能模块讲，是因为该模块涉及开发中一个重要的知识点。选择功能，界面实现效果如图 2-5 所示。

输入城市首字母，自动匹配城市。程序运行结果如图 2-6 所示。



图 2-5　手机订票系统城市选择界面



图 2-6　手机订票系统自动匹配城市界面

2.2.5　查询功能

查询功能是实现订票的具体业务功能。按照预订机票的传统系统，订票界面包括：出发城市的选择、到达城市的选择、航班的选择、出发时间、航班舱的选择。当取得网络连接，机票查询功能界面实现效果如图 2-7 所示，当未取得网络连接，机票查询界面实现效果如图 2-8 所示。在这里不一一把网络连接失败的效果图显示出来，读者需要注意的是涉及与网站后台进行通信的，都需要判断网络是否正常。这是处理通信异常的一种方法，将在项目的具体实现中一一讲述。



图 2-7　手机订票系统机票查询界面



图 2-8　网络连接失败不能订票



若网络没有连接上，不允许进行查票。怎样查看手机的网络是否打开，可以查看状态栏中的，有个“×”表示网络未连接上，需要打开网络连接才可以查票。

当选择城市按钮时，则弹出图 2-8 所示的城市选择界面；当选择出发时间时，弹出如图 2-9 所示的选择时间界面；当选择舱位时，弹出图 2-10 所示的选择舱位界面；当选择航空公司时，弹出图 2-11 所示的选择航空公司界面。



图 2-9　查询界面选择日期



图 2-10　查询界面选择舱位



图 2-11　查询界面选择航空公司

2.2.6　信息列表功能

手机订票系统严格遵循订票流程，当将出发城市、到达城市、航班的选择、出发时间及航班舱位等机票信息提交到数据后台时，要在界面上显示符合该查询线路的所有航班信息。手机订票系统的机票信息列表功能，程序运行结果如图 2-12 所示。



图 2-12　手机订票系统信息列表界面

2.2.7　List详情功能

由于机票系统 List 信息列表界面显示的信息有限，有必要将用户选择的每一条航班的具体信息更完全地显示出来，以便用户了解更详细的机票信息。手机订票系统的机票详情功能，程序运行结果如图 2-13 所示。



图 2-13　手机订票系统机票详情界面

2.2.8　用户信息填写与选择功能

用户查看航班的具体情况以后，要选择预订机票，那就需要填写个人信息（证件类型及订票人数）以便用户订票。手机订票系统预订功能，运行结果如图 2-14 所示。





图 2-14　手机订票系统个人信息填写与选择界面

2.2.9　预订成功功能

填写正确的订票信息后，将数据提交到后台，生成 PNR，将 PNR 返回，跳转到预订完成界面，此界面显示订票信息用户预订机票以后，要返回一个提示预订状态的信息。可在状态栏中添加一个消息通知，以便用户查看订票状态。手机订票系统的预订成功界面运行结果如图 2-15、图 2-16 所示。



图 2-15　手机订票系统预订成功界面

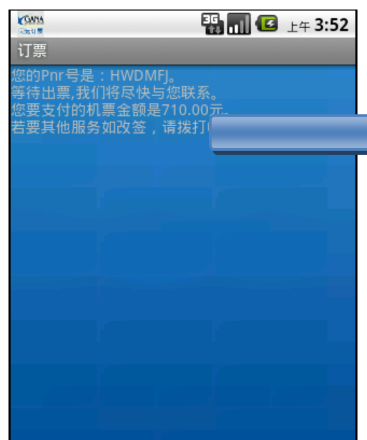


图 2-16　点开状态栏的消息提示，显示预订成功信息

2.3　项目后台技术选择——ASP.NET

Android 项目开发的整体实现，基本上都涉及与后台数据的交互。所有的手机开发工具

及使用到的开发语言（如 Android、Flex、Windows Mobile）都相当于实现手机应用程序的外观，虽然可以定义微型的数据库（SQLite），但是由于手机内存的局限性，通常采用一种后台技术来实现手机项目开发的后台。本项目就是使用 ASP 技术作为订票系统的后台支撑技术，之所以选择 ASP 技术，那是因为 ASP 是一种相对简单且稳定的，基于 IIS 的服务器端脚本编写环境，可以用来创建和运行动态网页或 Web 应用程序。下面简单介绍 ASP 技术的概况，由于本书的重点不是讲述 ASP 技术，在这里只对实现的原理以及项目中如何使用进行相应的讲解。如果读者未学习过 ASP 技术并对此技术感兴趣，可以阅读相关方面的书籍。

2.3.1 ASP.NET网页特点

ASP是Active Server Page的缩写，是动态服务器页面。它是微软公司开发的替代CGI（Common Gateway Interface）脚本程序的一种Web应用，它可以与数据库和其他程序进行交互，是一种相对简单、方便的编程工具。而ASP.NET是创建动态Web页的一种强大的服务器端新技术，它可为World Wide Web站点或企业内部互联网创建动态的可进行交互的HTML 页面，它采用面向对象的方法来构建动态 Web 应用程序，可在Internet或Intranet 上部署ASP.NET Web 应用程序。下面介绍ASP.NET的主要优点。

1）动态网页技术：利用 ASP.NET 可以突破静态网页的一些功能限制，实现动态网页技术。

2）更好的可扩展性：可在单独的机器或数据库的单独进程中维护会话状态，从而实现允许跨服务器的会话，且 ASP 文件是包含在 HTML 代码所组成的文件中，易于修改和测试。

3）增强服务器脚本处理功能：ASP.NET 提供了一些内置对象，使用这些对象可以使服务器端脚本功能更强。例如，可以从 Web 浏览器中获取用户通过 HTML 表单提交的信息，并在脚本中对这些信息进行验证及处理，然后向 Web 浏览器发送信息。

4）功能更为完善：ASP.NET 可以使用服务器端 ActiveX 组件来执行各种各样的任务，例如存取数据库、发送 Email 或访问文件系统等。

5）改进的安全性：ASP.NET 与 IIS、.NET 框架和操作系统所提供的基础安全服务配合使用，共同提供一系列身份验证和授权机制。由于服务器是将 ASP.NET 程序执行的结果以 HTML 格式传回客户端浏览器，因此使用者不会看到 ASP.NET 所编写的原始程序代码，可防止 ASP 程序代码被窃取。

6）配置和部署：将配置信息存储在基于 XML 的配置文件中，使得 ASP.NET 应用程序更易于部署。

7）支持多种编程语言开发：ASP.NET 支持 VB.NET、C#、Jscript 等多种语言的开发，不过目前使用较多的是 C#，本书也是使用 C#开发的后台。

2.3.2 ASP.NET 文件的体系结构

ASP.NET 的文件结构相对多且复杂，在表 2-1 中，列出了 ASP.NET 常见文件的主要功能，以便读者灵活使用各种类型的文件。



表 2-1　ASP.NET 文件的体系结构

文件扩展名	说　明	文件扩展名	说　明
.aspx	用于创建网页和对网页进行编程的核心文件类型	.axd	与 ASP.NET 应用程序跟踪有关联
.aspx.cs	由 ASPX 或 ASCX 文件继承的 C# 代码文件	.vsdisco	XML 文件，将链接公开给其他可描述 Web 服务的资源
.ascx	指明一个 ASP.NET 用户定义控件	.htm	标准 HTML 文件，包含静态元素和内容
.asax	包含 ASP.NET 应用程序级事件的事件语法	.xml	XML 文档，供 ASP.NET 应用程序使用
.asmx	供宿主 Web 服务在本地或远程使用	.config	配置文件，用于设置应用程序的各种属性

2.3.3　ASP.NET的工作原理

当在 Web 站点中融入 ASP.NET 功能后，将发生以下事情，也就是 ASP.NET 的工作原理，具体过程如图 2-17 所示。

1）客户请求 Web 页。

2）Web 服务寻找指令文件（.aspx）。

3）ASP.NET 代码被发送给公共语言运行时进行编译。

4）HTML 流返回给浏览器和指令。

5）浏览器处理 HTML 并显示页面。

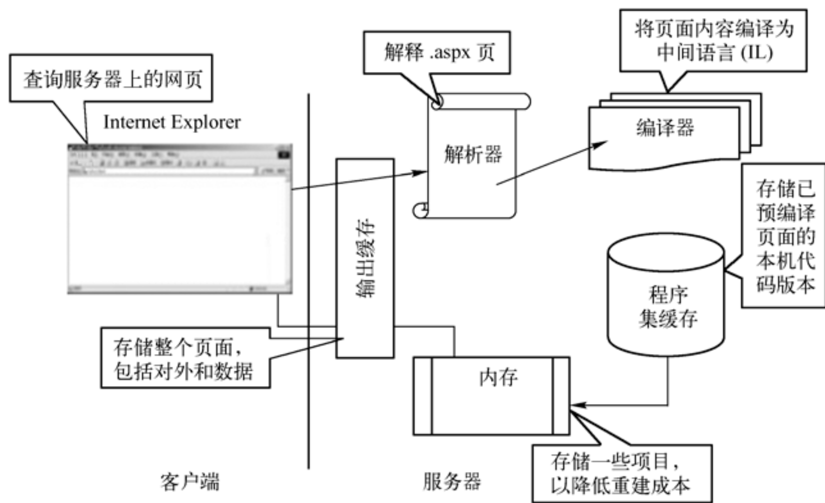


图 2-17　ASP.NET 工作原理图

2.3.4　IIS的安装与配置

在项目开发中，需要将我们编写的 ASP.NET 代码发布，然后部署到服务器地址下才能被 Android 前端项目所访问。在开发过程中，需要经常调整代码，因此有必要在自己的计算机上搭建一个 IIS 环境，如何部署 IIS、发布程序从而与 Android 应用程序进行通信，详细操作参考附录 A，在附录中详细讲述了 Windows XP 版和 Windows 7 版的 IIS 如何部署以及发布程序的过程。

第 3 章　程序主界面

手机订票系统程序主界面实现的结果图如第 2 章图 2-2 所示，功能主要涉及几个功能图标按钮的横向排列显示，当用户单击某一个图标时，跳转到相应的功能模块。要实现这一效果，可以使用 GridView 网格视图控件；若手机没有网络连接，需要使用消息提示功能提示不能进行相应的模块操作，这需要使用 Toast 消息提示控件。因此，在这一章的基础控件讲解部分，主要学习网格视图（GridView）和消息提示（Toast）的使用。

本章涉及的知识重点有两个：

1）基础数据适配器的使用，即如何为控件配置数据源。

2）Android 事件处理机制，即如何为控件设置事件监听。

3.1　基础控件讲解

基础控件是构成项目的灵魂，本章将针对项目界面涉及的控件逐一进行介绍。

3.1.1　网格视图

网格视图（GridView）的视图排列方式与矩阵类似，当屏幕上罗列的多个元素（如文字、图片等）同时显示的时候，可以考虑使用 GridView 控件。要掌握 GridView 的用法，关键是掌握如何使用基础数据适配器 BaseAdapter，它为 GridView 控件提供数据源。表 3-1 中列出 GridView 常用 xml 属性及方法。

表 3-1　GridView 常用 xml 属性及方法

GridView 常用 xml 属性及方法	描　　述
android:columnWidth	设置列的宽度
android:gravity	设置此组件中的内容在组件中的位置，即对齐方式。可选的值有：top、bottom、left、right、center_vertical、fill_vertical、center_horizontal、fill_horizontal、center、fill、clip_vertical 可以多选，用“|”分开
android:horizontalSpacing	两列之间的间距
android:numColumns	每一行显示的列数，可以根据内容自动调整列数，设置方法是：android:numColumns="auto_fit"
android:stretchMode	缩放模式，可选值有：columnWidth，spacingWidth，spacingWidthUniform、none
android:verticalSpacing	两行之间的间距
getAdapter ()	获得与此组件相关的适配器
setOnItemClickListener()	对 GridView 的子项进行监听，以便对子项进行具体操作

下面通过一个示例讲述 GridView 的用法，主要功能是有规律地显示 7 张图片按钮，且捕获了各个子项按钮的鼠标单击事件，运行结果如图 3-1 所示，当用户单击某一张图片时，处理事件如图 3-2 所示。在这里只给出 main.xml 和 MainActivity.java 中的代码，默认



AndroidManifest.xml 中的代码没有做任何变化，如果需要在此配置文件中增加相应的权限，会具体给出相应的代码。



图 3-1　GridView 网格视图示例代码运行结果（1）——初始化



图 3-2　GridView 网格视图示例代码运行结果（2）——事件处理

首先，介绍如何通过 xml 布局实现网格视图这一效果，如代码清单 3-1 所示。main.xml 文件在项目中的具体路径位置是：第 3 章\ Demo_03_01\ res\layout\ main.xml，其他示例的 xml 布局文件的位置类似。

代码清单 3-1　GridView网格视图示例（第 3 章\ Demo_03_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridView
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/gridview"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:numColumns="auto_fit"
     android:verticalSpacing="10dp"
     android:horizontalSpacing="10dp"
     android:columnWidth="90dp"
     android:stretchMode="columnWidth"
     android:gravity="center"
/>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现功能效果，在这里定义了一个 ImageAdapter，它继承 BaseAdapter，用来存储要显示的图片，关于 BaseAdapter 的进一步讲解将在重点剖析中讲解，实现这一过程如代码清单 3-2 所示。要捕获到用户单击了某一张图片按钮，那就要监听 GridView 单击事件，实现 onItemClick 方法，捕获子项 Id（position）具体代码实现过程如代码清单 3-3 所示。ImageAdapter.java 和 MainActivity.java 文件在项目中的具体路径位置是：第 3 章\ Demo_03_01 \src\com\chen\android\文件夹下。

代码清单 3-2 GridView网格视图示例（第 3 章\Demo_03_01）ImageAdapter.java

```
package com.chen.android;
import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;
/**
 * GridView 数据源
 */
public class ImageAdapter extends BaseAdapter
{
	// 定义 Context 上下文，在本例中是指 MainActivity.java
	private Context           mContext;
	// 定义整型数组 即图片数据源
	private Integer[] mImages    =
	{
			R.drawable.img1,
			R.drawable.img2,
			R.drawable.img3,
			R.drawable.img4,
			R.drawable.img5,
			R.drawable.img6,
			R.drawable.img7,
	};
	//构造函数
	public ImageAdapter(Context c)
	{
		mContext = c;
	}
	// 获取图片的个数
	public int getCount()
	{
		return mImages.length;
	}
	// 获取图片在库中的位置
	public Object getItem(int position)
	{
		return position;
	}
	// 获取图片 ID
	public long getItemId(int position)
	{
		return position;
	}
	/**
	 * 当列表里的每一项显示到页面时，都会调用 Adapter 的 getView 方法返回一个 View
	 */
	public View getView(int position, View convertView, ViewGroup parent)
```



```
        {
            ImageView imageView;
            if (convertView == null)
            {
                // 给 ImageView 设置资源
                imageView = new ImageView(mContext);
                // 设置布局 图片 85×85 显示
                imageView.setLayoutParams(new GridView.LayoutParams(85, 85));
                // 设置显示比例类型（（详见 ImageView 的 ScaleType 值含义说明））
                imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
            }
            else
            {
                imageView = (ImageView) convertView;
            }
            imageView.setImageResource(mImages[position]);
            return imageView;
        }
    }
```

代码清单 3-3　GridView网格视图示例（第 3 章\Demo_03_01）MainActivity.java

```
    package com.chen.android;
    import android.app.Activity;
    import android.os.Bundle;
    import android.view.Gravity;
    import android.view.View;
    import android.widget.AdapterView;
    import android.widget.GridView;
    import android.widget.Toast;
    import android.widget.AdapterView.OnItemClickListener;
    /**
     * GridView 网格视图示例，继承 Activity
     * @author
     */
    public class MainActivity extends Activity
    {
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            //设置一个 Activity 的显示界面，显示界面在 res 文件夹下的 layout 中
            setContentView(R.layout.main);
            //取得 GridView 对象
            GridView gridview = (GridView) findViewById(R.id.gridview);
            //添加元素给 gridview
            gridview.setAdapter(new ImageAdapter(this));
            // 设置 Gallery 的背景
            gridview.setBackgroundResource(R.drawable.bg0);
            //事件监听
            gridview.setOnItemClickListener(new OnItemClickListener() {
```

```
                public void onItemClick(AdapterView<?> parent, View v, int position, long id)
                {
                    DisplayToast("您单击了" + (position + 1) + " 号图片");
                }
            });
        }

        /* 显示 Toast */
        public void DisplayToast(String str) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            // 设置 toast 显示的位置
            toast.setGravity(Gravity.TOP, 0, 180);
            // 显示该 toast
            toast.show();
        }
    }
```

3.1.2　提示

几乎所有的手机应用程序开发都有消息提示框，Android 也不例外，也有消息提示（Toast）功能，它通过 Toast 组件来实现。Toast 实现的效果是在手机屏幕上显示一条信息，显示一段时间后信息会自动消失。信息可以是简单的文本，也可以是复杂的图片及其他内容（可以显示一个 view）。Toast 消息提示常用方法如表 3-2 所示。

表 3-2　Toast 消息提示常用方法

Toast 消息提示	功 能 说 明
Toast.makeText(this, " 消息提示的内容", Toast.LENGTH_LONG);	在 Activity 中构造出一个 Toast，并设置消息提示的内容
setView()	设置 Toast 要显示的画面，一般是 xml 布局画面
setDuration()	设置显示时间，为长时间 Toast.LENGTH_LONG，短时间为 Toast.LENGTH_SHORT
Show()	显示所定义的 Toast 控件

下面通过一个示例讲述 Toast 的具体用法，该示例的主要功能是提供两种方式显示 Toast，一种是文本形式的 Toast 信息提示，另一种是由文本和图片组成的 Toast 信息提示，代码运行结果如图 3-3、图 3-4 所示。

首先，介绍如何通过 xml 布局实现 Toast 消息提示功能，如代码清单 3-4、代码清单 3-5 所示。



图 3-3　Toast 消息提示示例（1）



图 3-4　Toast 消息提示示例（2）



代码清单 3-4　Toast消息提示示例（第 3 章\Demo_03_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >
       <Button android:id="@+id/button1"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Toast 显示 View"
       />
       <Button android:id="@+id/button2"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="Toast 直接输出"
       />
</LinearLayout>
```

代码清单 3-5　Toast消息提示示例（第 3 章\Demo_03_02）toast.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >
       <ImageView android:src="@drawable/icon"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
       />
       <TextView android:id="@+id/tv1"
             android:text=""
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
       />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 Toast 消息提示。在实现过程中，需要注意 LayoutInflater 的使用，它的主要功能是将 Layout 中的 xml 文件转换为 View，它是专门供 Layout 使用的 Inflater。虽然 Layout 也是 View 的子类，但在 Android 中如果想将 xml 文件中的 Layout 转换为 View 放入.java 代码中操作，只能通过 Inflater，而不能通过 findViewById()，如代码清单 3-6 所示。

代码清单 3-6　Toast消息提示示例（第 3 章\Demo_03_02）MainActivity.java

```
package com.chen.android;
```




```
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.view.Gravity;
import android.view.LayoutInflater;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

/**
 *   Toast 信息提示的使用
 *
 * @author
 */
public class MainActivity extends Activity {
	/** 这是程序入口 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		/* 获得 Button1 对象 */
		Button button1 = (Button) findViewById(R.id.button1);
		/* 设置 button1 的事件监听 */
		button1.setOnClickListener(button1Toast);
		/* 获得 Button2 对象 */
		Button button2 = (Button) findViewById(R.id.button2);
		/* 设置 button2 的事件监听 */
		button2.setOnClickListener(button2Toast);
	}

	/*
	 * 实现监听
	 */
	OnClickListener button1Toast = new OnClickListener() {
		@Override
		public void onClick(View v) {
			/* toast 显示信息 */
			showToast();
		}
	};

	/*
	 * 实现监听
	 */
	OnClickListener button2Toast = new OnClickListener() {
		@Override
		public void onClick(View v) {
			/* toast 显示信息 */
```



```
                    DisplayToast("直接输出信息");
                }
            };

            /**
             * 嵌套 toast.xml 的 Toast 显示信息
             */
        public void showToast() {
             LayoutInflater layoutInflater
        = (LayoutInflater) getSystemService(Context.LAYOUT_INFLATER_SERVICE);
                //把 xml 表述的 layout 转化为 View
                View view = layoutInflater.inflate(R.layout.toast, null);
                // 把布局文件 toast.xml 转换成一个 view
                Toast toast = new Toast(this);
        // 载入 view,即显示 toast.xml 的内容
                toast.setView(view);

                TextView textView1 = (TextView) view.findViewById(R.id.tv1);
                // 修改 TextView 里的内容
                textView1.setText("Toast 显示 toast.xml 中的内容");
            // 设置显示时间，长时间 Toast.LENGTH_LONG，短时间为 Toast.LENGTH_SHORT
                toast.setDuration(Toast.LENGTH_SHORT);
                //显示 Toast
                toast.show();
            }

            /**
             * Toast 提示框
             */
            public void DisplayToast(String str) {
                /* 弹出信息，可设置弹出信息的长短，不可编辑时间长短 */
                Toast toast = Toast.makeText(this, str, Toast.LENGTH_LONG);
                // 设置 toast 显示的位置（x,y）
                toast.setGravity(Gravity.TOP, 0, 220);
                // 显示该 toast
                toast.show();
            }
        }
```

3.2　重点剖析

重点剖析介绍功能界面实现中需要重点掌握的 Android 编程方法与编程思路，本节主要介绍数据适配器 Adapter、Android 的事件处理机制、Android 页面切换等内容。

3.2.1　数据适配器

本节重点讲解数据适配器（Adapter），Adapter 的作用就是为 GridView、ListView 等界面控件与数据之间搭起一座桥梁，当列表里的每一项显示到页面时，都会调用 Adapter 的 getView 方法返回一个 View。在 Android API 中内置了几个实现了 ListAdapter 的 Adapter，

它们分别是：BaseAdapter，SimpleAdapter（以 Map 的形式存储静态数据，可多态实现 ListAdapter），SimpleCursorAdapter 用于游标的形式查询结果）等。Adapter 在 Android 应用程序开发中起着非常重要的作用，应用也非常广泛，它可看做是数据源和 UI 组件之间的桥梁，其中 Adapter、数据和 UI 三者之间的关系，如图 3-5 所示。

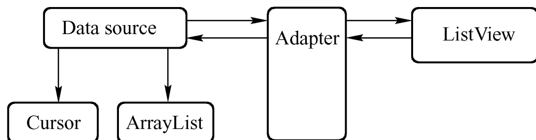


图 3-5　Adapter、数据和 UI 三者之间的关系

而 Adapter 包含的子类有：BaseAdapter、ArrayAdapter 及 SimpleCusorAdapter 等。BaseAdapter 是抽象类，实现了 ListAdapter 和 SpinnerAdapter 两个接口，当然它也可以直接给 ListView 和 Spinner 等 UI 组件直接提供数据。ArrayAdapter 是从 BaseAdapter 派生出来的，具备 BaseAdapter 的所有功能，但 ArrayAdapter 更为强大，它实例化时可以直接使用泛型构造支持泛型操作。通常需要实现 getView()方法，为了便于数据处理，最好要重写 getItemId()。SimpleCusorAdapter 可以适用于简单的纯文字型 ListView，它需要 Cursor 的字段和 UI 控件的 Id 对应起来，如果要实现更复杂的 UI 也可以重写其他方法。

通常我们更多地继承 BaseAdapter 来编写自己的 Adapter 类， 因为 BaseAdapter 类是其他 Apdater 类的基类。在实际运用过程中，扩展 BaseAdapter 类一般都需要重写表 3-3 所示的方法。

表 3-3　自定义 BaseAdapter 需要重写的方法

返 回 值	方 法 名	功 能 说 明
int	getCount()	获取当前 Adapter 的 Items 数目
Object	getItem(int position)	获取相应 position 的 Item
long	getItemId(int position)	获取相应 position 的 Item 在 List 中的 row id
View	getView(int position，View convertView， ViewGroup parent)	获取在指定 position 所要显示的 data 的 View

这几个需要实现的方法在 Adapter 类中的结构如图 3-6 所示，除了上述表中列出的方法，在 Adapter 类中还定义了许多方法，用来捕获数据源相关属性特征，以便操作该子项。如定义了可以通过数据源子项 Id 查找到该子项，获取数据源子项个数，判断数据源是否为空等方法。

自定义 Adapter 子类，就需要重写这几个方法，在这里进一步讲解 getView()方法，它在 Adapter 类起到举足轻重的作用，主要是将获取数据后的 View 组件返回。比如，可以在此方法中读取到所定义的 GridView 中的每一行里 ImageView 传递过来的属性及数据，还可以读取到 ListView 中每一行里的 TextView 传递过来的属性及数据。

下面通过一个示例讲述 Android 适配器的使用方法。该示例的主要功能是在 Gallery 中显示一组图片，效果如图 3-7 所示。在功能中分两行显示，上一行是显示一组图片，下一行是当单击某一张图片的时候，放大选中的图片。

下面介绍如何通过 xml 布局实现 Adapter 显示图片组功能，如代码清单 3-7 至代码清单 3-8 所示。



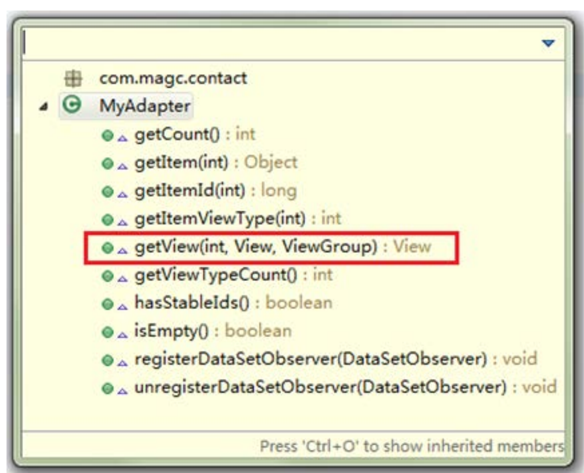


图 3-6　Adapter 结构



图 3-7　Adapter 显示图片组示例代码运行结果

代码清单 3-7　Adapter显示图片组示例（第 3 章\Demo_03_03）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/hello"
         />
     <Gallery
         android:id="@+id/gallery1"
         android:layout_width="match_parent"
         android:spacing="5px"
         android:layout_height="wrap_content"
     />
     <ImageView
         android:id="@+id/imageView1"
         android:layout_gravity="center_vertical"
         android:layout_marginTop="20px"
         android:layout_width="320px"
         android:layout_height="320px"
         />
</LinearLayout>
```

代码清单 3-8　Adapter显示图片组示例（第 3 章\Demo_03_03）MainActivity.java

```
package com.chen.android;
```




```
import android.app.Activity;
import android.content.Context;
import android.os.Bundle;
import android.util.Log;
import android.view.View;
import android.view.ViewGroup;
import android.widget.AdapterView;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.AdapterView.OnItemClickListener;

/**
 *Adpter 的使用方法
 * @author
 *
 */
public class MainActivity extends Activity {
 /*定义 Gallery（画廊功能）和 ImageView（显示图片）组件，将在后续章节中介绍*/
    private Gallery gallery;
    private ImageView imgview;
    /*定义数据源*/
    private int[] imgs = {R.drawable.f4,R.drawable.f1,R.drawable.f2,R.drawable.f3};
    /** 当 activity 首次创建时响应调用 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
      //设置一个 Activity 的显示界面，显示界面在 res 文件夹下的 layout 中
        setContentView(R.layout.main);
        imgview = (ImageView)findViewById(R.id.imageView1);
        gallery = (Gallery)findViewById(R.id.gallery1);
        MyImgAdapter adapter = new MyImgAdapter(this);
        gallery.setAdapter(adapter);
        gallery.setOnItemClickListener(new OnItemClickListener() {
            //用户单击图片时，将该图片的 ResourceID 传递到下面的 ImageView 中
            @Override
            public void onItemClick(AdapterView<?> arg0, View view, int position,
                    long arg3) {

                imgview.setImageResource(imgs[position]);

            }
        });
    }
    /**
     * 自定义图片 Adapter 以内部类形式存在于 MainActivity 中，
     * 方便访问 MainActivity 中的各个变量，特别是 imgs 数组
     */
     class MyImgAdapter extends BaseAdapter {
    /* 定义 Context 上下文，在本例中是指 MainActivity.java*/
```



```
        private Context context;
        public MyImgAdapter(Context context) {
            super();
            this.context = context;
        }
      //  获取图片的个数
        @Override
        public int getCount() {
            return imgs.length;
        }
      //  获取图片在库中的位置
        @Override
        public Object getItem(int position) {
            return position;
        }
       //  获取图片 ID
        @Override
        public long getItemId(int position) {
            return position;
        }

       /**
       *  当列表里的每一项显示到页面时，
 *  都会调用 Adapter 的 getView 方法返回一个 View
       */
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {
            //针对每一个数据（即每一个图片 ID）创建一个 ImageView 实例
            ImageView imageView = new ImageView(context);
            //设置图片数据源
            imageView.setImageResource(imgs[position]);
            //写日志
            Log.i("magc", String.valueOf(imgs[position]));
          //设置 Gallery 中每一个图片的大小为 80*80
            imageView.setLayoutParams(new Gallery.LayoutParams(80, 80));
            //  设置显示比例类型
            imageView.setScaleType(ImageView.ScaleType.FIT_XY);
            return imageView;
        }

    }
}
```

3.2.2　Android事件处理机制

当今所有的用户界面，都是以事件驱动来实现人机交互的过程，而 Android 的事件处理机制就是用户与图形界面之间进行交互时所触发的操作。例如，在用户按下某一个按钮（Button）时，就会触发单击事件及触摸屏事件，这时我们可以监听这些事件并进行相应的操作，表 3-4 中列出了常用事件的功能清单。

表 3-4　Android 常用事件功能清单

事　　件	功 能 说 明
onClick()	包含于 View.OnClickListener 中。当用户触摸到当前的这个 item，或者通过浏览键或跟踪球聚焦在这个 item 上，然后按下"确认"键或者按下跟踪球时被调用。通俗讲就是当用户单击某一个控件，这个控件实现了对该事件的监听，那么这个 item 就会被触发，响应时间比较短
onLongClick()	包含于 View.OnLongClickListener 中。当用户触摸并控制住当前的这个 item，或者通过浏览键或跟踪球聚焦在这个 item 上，然后保持按下"确认"键或者按下跟踪球（一秒钟）时被调用。与 onClick()用法类似，只是响应时间相对较长，一秒钟左右
onFocusChange()	包含于 View.OnFocusChangeListener 中。当用户使用浏览键或跟踪球浏览进入或离开当前的这个 item 时被调用
onKey()	包含于 View.OnKeyListener 中。当用户聚焦在当前的这个 item 上并按下或释放设备上的一个按键时被调用。包括：onKeyDown（用户响应按钮按下）、onKeyUp（用户相应按钮释放）、onKeyMultiple（用于响应按钮重复单击）等
onTouch()	包含于 View.OnTouchListener 中。当用户执行的动作被当做是一个触摸事件时被调用，包括按下、释放，或者屏幕上任何的移动手势（在这个 item 的边界内）
onCreateContextMenu()	包含于 View.OnCreateContextMenuListener 中。当正在创建一个上下文菜单的时候被调用

下面通过一个示例讲述 Android 事件处理机制的用法，该示例的主要功能如下。

1）单击 onClick 按钮时，捕获的是该按钮的单击事件，并通过 DisplayToast 显示简单的文本信息；触摸屏单击 onTouch 按钮时，捕获的是该按钮的触摸屏事件，并通过 DisplayToast 显示捕获到的事件。

2）键盘按键被按下和弹起时所触发的事件（上、中、下、左、右）。

3）实现了触摸屏事件，可以捕获屏幕上的某一点的坐标。

示例代码运行结果如图 3-8、图 3-9、图 3-10 及图 3-11 所示。



图 3-8　Android 事件处理示例代码运行结果（1）——单击 OnClick 按钮



图 3-9　Android 事件处理示例代码运行结果（2）——触摸屏单击 OnTouch 按钮



图 3-10　Android 事件处理示例代码运行结果（3）——单击屏幕触发触摸屏事件



图 3-11　Android 事件处理示例代码运行结果（3）——按下手机键盘上方向键

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 3-9 所示。

代码清单 3-9　Android 事件处理示例（第 3 章\Demo_03_04）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```



```
<LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
         <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="@string/hello"
        />
        <LinearLayout
          android:layout_width="fill_parent"
          android:layout_height="fill_parent"
          android:orientation="horizontal">
                <Button
                        android:id="@+id/btn_confirm"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="OnClick 事件"
                        />
                        <Button
                        android:id="@+id/btn_cancel"
                        android:layout_width="wrap_content"
                        android:layout_height="wrap_content"
                        android:text="OnTouch 事件"
                        />
          </LinearLayout>

</LinearLayout>
```

接着，介绍 Activity java 代码如何配合 xml 布局实现这一效果，此处监听了按钮的单击、触摸屏事件，并且重写了 onKeyDown()方法，如代码清单 3-10 所示。

代码清单 3-10　Android 事件处理示例（第 3 章\Demo_03_04）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.KeyEvent;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.Toast;
/**
  * Android 事件处理机制
  * @author
  */
public class MainActivity extends Activity {
```




```
/*初始化控件*/
    Button button_confirm;
    Button btn_cancel;
    public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            // 获得 Button 对象
             button_confirm = (Button) findViewById(R.id.btn_confirm);
             btn_cancel= (Button) findViewById(R.id.btn_cancel);
            // 设置 Button 控件监听器
            button_confirm.setOnClickListener(btnClickConfirmClick);

            btn_cancel.setOnTouchListener(btnTouchClick);
    }

    /**
      * 实现 onClick 按钮的监听
      */
Button.OnClickListener btnClickConfirmClick = new Button.OnClickListener() {

            @Override
            public void onClick(View v) {
                    // TODO Auto-generated method stub
                    // 这里处理事件
                    DisplayToast("单击了确定按钮");
            }
    };

    /**
      * 实现 onTouch 按钮的监听
      */
Button.OnTouchListener btnTouchClick = new Button.OnTouchListener() {

    @Override
    public boolean onTouch(View arg0, MotionEvent event) {
            // 这里处理事件
            int iAction = event.getAction();
            String iActionStr;
            switch (iAction) {
            case MotionEvent.ACTION_DOWN:
                    iActionStr="ACTION_DOWN";
                    break;
            case MotionEvent.ACTION_UP:
                    iActionStr="ACTION_UP";
                    break;
            case MotionEvent.ACTION_CANCEL:
                    iActionStr="ACTION_CANCEL";
                    break;
            case MotionEvent.ACTION_MOVE:
                    iActionStr="ACTION_MOVE";
                    break;
```



```
            default:
                iActionStr="其他键";
                break;
            }
            DisplayToast("触发了"+iActionStr+"事件");
            return false;
        }
    };

    /* 按键被按下所触发的事件 */
    public boolean onKeyDown(int keyCode, KeyEvent event) {
            switch (keyCode) {
            case KeyEvent.KEYCODE_DPAD_UP:
                DisplayToast("按下：上方键");
                break;
            case KeyEvent.KEYCODE_DPAD_CENTER:
                DisplayToast("按下：中键");
                break;
            case KeyEvent.KEYCODE_DPAD_DOWN:
                DisplayToast("按下：下方键");
                break;
            case KeyEvent.KEYCODE_DPAD_LEFT:
                DisplayToast("按下：左方键");
                break;
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                DisplayToast("按下：右方键");
                break;
            }
            return super.onKeyDown(keyCode, event);
    }

    /* 按键弹起所触发的事件 */
    public boolean onKeyUp(int keyCode, KeyEvent event) {
            switch (keyCode) {
            case KeyEvent.KEYCODE_DPAD_UP:
                DisplayToast("弹起：上方键");
                break;
            case KeyEvent.KEYCODE_DPAD_CENTER:
                DisplayToast("弹起：中键");
                break;
            case KeyEvent.KEYCODE_DPAD_DOWN:
                DisplayToast("弹起：下方键");
                break;
            case KeyEvent.KEYCODE_DPAD_LEFT:
                DisplayToast("弹起：左方键");
                break;
            case KeyEvent.KEYCODE_DPAD_RIGHT:
                DisplayToast("弹起：右方键");
                break;
            }
```

```
            return super.onKeyUp(keyCode, event);
        }

    /*按钮重复单击*/
        public boolean onKeyMultiple(int keyCode, int repeatCount, KeyEvent event) {
            return super.onKeyMultiple(keyCode, repeatCount, event);
        }

        /* 触摸屏事件 */
        public boolean onTouchEvent(MotionEvent event) {
            int iAction = event.getAction();
            if (iAction == MotionEvent.ACTION_CANCEL
                    || iAction == MotionEvent.ACTION_DOWN
                    || iAction == MotionEvent.ACTION_MOVE) {
                return false;
            }
            // 得到触笔单击的位置
            int x = (int) event.getX();
            int y = (int) event.getY();

            DisplayToast("触笔单击坐标：(" + Integer.toString(x) + ","
                    + Integer.toString(y) + ")");

            return super.onTouchEvent(event);
        }

        /* 显示 Toast */
        public void DisplayToast(String str) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            // 设置 toast 显示的位置
            toast.setGravity(Gravity.TOP, 0, 150);
            // 显示该 toast
            toast.show();
        }
}
```

3.2.3 Android页面切换

在 Android 中如何实现两个页面之间的切换呢？它不像网络编程中的 HTML 那样，使用一个超链接标签（<a>）实现切换，Android 中实现页面的切换要使用到 Intent 类，在 Android 参考文档中，对 Intent 的定义是执行某操作的一个抽象描述。该概念显得相对抽象，通俗地讲，Intent 主要是在应用程序传值中起到媒介的作用，专门提供组件互相调用的相关信息，实现调用者与被调用者之间的解耦。Intent 还负责对应用中的一次操作的动作、动作中涉及的数据进行描述，Android 程序则会根据此 Intent 描述，负责查找对应的组件，并将 Intent 负载的信息传递给调用的组件，从而完成组件的调用。

下面通过一个示例讲述 Android 页面切换的用法，该示例的主要功能是：单击按钮切换到第二个页面。代码运行结果如图 3-12 和图 3-13 所示。





图 3-12　Android 页面切换示例代码运行结果——初始化界面



图 3-13　Android 页面切换示例代码运行结果——切换界面

首先，介绍如何通过 xml 布局实现这一效果，在示例中定义了两个 xml 布局。如代码清单 3-11、3-12 所示。

代码清单 3-11　Android页面切换示例（第 3 章\Demo_03_05）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

   <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="这是一个页面"
    />
    <Button
        android:id="@+id/btnSend"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="跳转到第二个页面"
        android:gravity = "center"
     />
</LinearLayout>
```

代码清单 3-12　Android页面切换示例（第 3 章\Demo_03_05）sencond.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">

     <TextView
         android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:text="这是第二个页面" />
     <Button
        android:id="@+id/btnReturn"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="返回"
```

```
            android:gravity = "center"
            />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，此处监听了按钮的单击事件，如代码清单 3-13、3-14 所示。

代码清单 3-13　Android页面切换示例（第 3 章\Demo_03_05）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
/**
 * 这是 Demo_03_05,实现页面切换
 * @author
 */
public class MainActivity extends Activity {

     /* 声明 Button 对象 */
     private Button    m_Send;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*获取 main.xml 中的 button*/
        m_Send=(Button)findViewById(R.id.btnSend);
        /*设置事件监听*/
        m_Send.setOnTouchListener(Onsent);
    }

    /**
     * button 事件实现
     */
    Button.OnTouchListener    Onsent=new    Button.OnTouchListener(){
           @Override
           public boolean onTouch(View arg0, MotionEvent arg1) {
                /*页面跳转*/
                /* 新建一个 Intent 对象 */
                Intent intent = new Intent();
                /* 指定 intent 要启动的类 */
                intent.setClass(MainActivity.this, Sencond.class);
                /* 启动一个新的 Activity */
                startActivity(intent);
                /* 关闭当前的 Activity */
                MainActivity.this.finish();
                return false;
           }
    };
}
```



代码清单 3-14　Android页面切换示例（第 3 章\Demo_03_05）Sencond.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
/**
 * 这是 Demo_03_05,实现页面的跳转
 * @author
 */
public class MainActivity extends Activity {

     /* 声明 Button 对象 */
     private Button   m_Send;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*获取 main.xml 中的 button*/
        m_Send=(Button)findViewById(R.id.btnSend);
        /*设置事件监听*/
        m_Send.setOnTouchListener(Onsent);
    }

    /**
     * button 事件实现
     */
    Button.OnTouchListener   Onsent=new   Button.OnTouchListener(){
          @Override
          public boolean onTouch(View arg0, MotionEvent arg1) {
               /*页面跳转*/
               /* 新建一个 Intent 对象 */
               Intent intent = new Intent();
               /* 指定 intent 要启动的类 */
               intent.setClass(MainActivity.this, Sencond.class);
               /* 启动一个新的 Activity */
               startActivity(intent);
               /* 关闭当前的 Activity */
               MainActivity.this.finish();
               return false;
          }
    };
}
```

最后，需要在 AndroidManifest.xml 配置文件中添加需要跳转的页面，具体如代码清单 3-15 所示。

代码清单 3-15　Android页面切换示例（第 3 章\Demo_03_05）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```




```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Sencond"></activity>
    </application>
</manifest>
```

3.3 程序主界面功能实现

通过前面章节详细讲解项目主界面涉及的相关控件的使用以及对相关重点进行剖析，相信大家对实现手机订票系统的主界面有信心了吧，那就赶快行动吧。

从这一章节开始，就要系统地学习手机订票系统。首先要建项目，可以按照第 1 章 HelloAndroid 所讲的步骤，开始开发手机订票系统这一项目，项目的相关设置如图 3-14 所示。

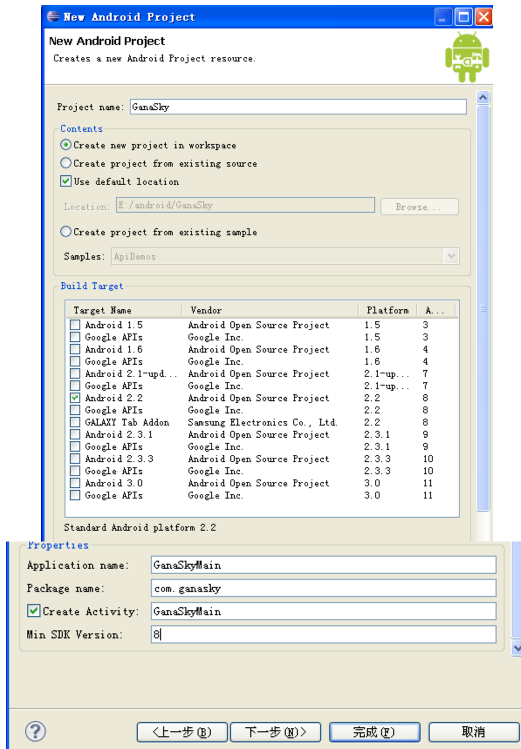


图 3-14　创建手机订票系统项目并设置相关参数



除了建 com.ganasky 包，还需建 com.ganasky.Help、com.ganasky.model 包，Activity java 代码放在 com.ganasky 包里，其他的包将在后面的章节中用到。整个项目的结构如图 3-15 所示。

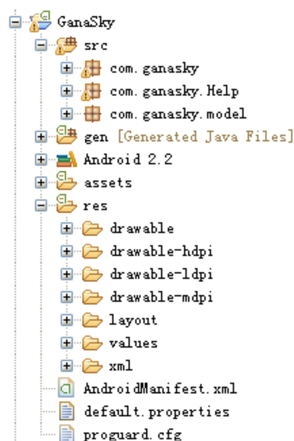


图 3-15 项目结构图

3.3.1 程序主界面View实现

主界面的 View 实现，其实就是 layout 下的 xml 外观布局实现，之所以取名为 View 是想强调 Android 三层架构的开发思想，相信有过软件开发经验的人对该架构非常熟悉，它是开发中使用最普遍的编程思想。根据第 2 章手机订票系统实现效果图（图 2-2）所示，界面涉及 GridView 控件，主要是巩固该控件的使用。相信经过上一阶段的讲解，实现订票系统的主界面外观样式相对容易，如代码清单 3-16 所示。

代码清单 3-16 订票系统主界面（第 3 章\GanaSkyForMain）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<GridView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/gridview"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:numColumns="auto_fit"
    android:verticalSpacing="10dp"
    android:horizontalSpacing="10dp"
    android:columnWidth="90dp"
    android:stretchMode="columnWidth"
    android:gravity="center"
/>
```

3.3.2 程序主界面Control实现

主界面的 Control 实现，其实就是 src 下的 Activity java 代码，之所以取名为 Control，也是为了强调 Android 三层架构开发思想，和 View 配套对应。具体代码实现如代码清单 3-17 至代码清单 3-20 所示，在实现过程中需要注意的有以下几点。

1）我们开发的是手机应用项目，且触屏手机越来越普及，因此，在选择事件处理类型的时候，可以选择触屏事件。程序主界面使用到 GridView 的子项单击事件，它具备子项触屏单击的功能，因此可以监听 GridView 的 onItemClick 事件。在实现 onItemClick 的方法中，要捕获到 position 的值。根据效果图，第一项是登录界面，第二项是查票主界面，第三项是退出界面。在这里可以先建 LoginApp.java 和 GanaSkyMain.java 类，具体实现将在后面的章节中涉及。

2）关于如何跳转页面的问题。在第 1 章剖析 HelloAndroid 的结构中已经谈及过，Android 中的每一个可显示的页面，都要继承 Activity，且要在 AndroidManifest.xml 中添加 activity 节点，如在 AndroidManifest.xml 中添加 <activity android:name="LoginApp"

android:label="@string/app_nameLogin"/>才能实现主界面切换到登录页面。

3）不管是登录还是查票界面，都涉及如何与后台交互，因此要在保证网络畅通的情况下才能开启系统功能，这需要在 ConnUrlHelper.hasInternet()方法中判断网络是否畅通，还要在 AndroidManifest.xml 文件中添加访问权限。

4）退出系统功能也是主界面实现的重点，需要注意的也是要在 AndroidManifest.xml 配置文件中添加 android.permission.RESTART_PACKAGES 访问权限。

代码清单 3-17 订票系统主界面实现（第 3 章\GanaSkyForMain）GanaSkyMain.java

```
package com.ganasky;

import com.ganasky.Help.ConnUrlHelper;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.AdapterView;
import android.widget.AdapterView.OnItemClickListener;
import android.widget.GridView;
import android.widget.Toast;

/**
 * 这是主页面
 * @author
 *
 */
public class GanaSkyMain extends Activity {
	private int item = 0;
	private GanaSkyImageAdapter imageApter;

	/** 这是 ganaSky 的主页面 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		// 加载主页面
		setContentView(R.layout.main);

		// 取得 GridView 对象
		GridView gridview = (GridView) findViewById(R.id.gridview);

		// 初始化图片容器对象
		imageApter = new GanaSkyImageAdapter(this);

		// 给 gridview 控件添加元素
		gridview.setAdapter(imageApter);

		// 设置订票系统的背景
		gridview.setBackgroundResource(R.drawable.bg0);
```



```
        // 事件监听
        gridview.setOnItemClickListener(new OnItemClickListener() {
            @Override
            public void onItemClick(AdapterView<?> parent, View v,
                    int position, long id) {
                item = position + 1;
                switch (item) {
                 /*跳转到登录界面*/
                case 1:
                    /* 新建一个 Intent 对象 */
                    Intent intent = new Intent();
                 /* 指定 intent 要启动的类，可以先建 LoginApp.java 文件，
                 * 在后面的章节中再实现 */
                    intent.setClass(GanaSkyMain.this, LoginApp.class);
                    /* 启动一个新的 Activity */
                    startActivity(intent);
                    /* 关闭当前的 Activity */
                    GanaSkyMain.this.finish();
                    break;

                 /*跳转到查票主界面*/
                case 2:
                    /*判断是否有网络接入*/
                    if (ConnUrlHelper.hasInternet(GanaSkyMain.this)) {
                        /* 新建一个 Intent 对象 */
                        Intent intent2 = new Intent();
                        /* 指定 intent 要启动的类 */
                        intent2.setClass(GanaSkyMain.this, TicketMainApp.class);
                        /* 启动一个新的 Activity */
                        startActivity(intent2);
                        /* 关闭当前的 Activity */
                        GanaSkyMain.this.finish();
                    } else {
                        DisplayToast("网络连接失败,不能订票！");
                    }
                    break;
                case 3:
                    /*退出系统*/
                    exitApp2();
                    break;
                }
            }
        });
    }

    /**
     * 退出系统
     */
    private void exitApp2() {
```

```
            // 先溢出用户信息
            Intent startMain = new Intent(Intent.ACTION_MAIN);
            startMain.addCategory(Intent.CATEGORY_HOME);
            startMain.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(startMain);
            //退出系统
            System.exit(0);
        }

        /**
         * Toast 提示框
         */
        public void DisplayToast(String str) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            // 设置 toast 显示的位置
            toast.setGravity(Gravity.TOP, 0, 220);
            // 显示该 toast
            toast.show();
        }

}
```

代码清单 3-18　订票系统主界面GanaSkyImageAdapter.java实现

```
package com.ganasky;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.GridView;
import android.widget.ImageView;

/**
 * 主界面数据适配器实现
 * @author
 */
public class GanaSkyImageAdapter extends BaseAdapter
{
    // 定义 Context
    private Context          mContext;
    // 定义整型数组 即图片源
    private Integer[] mImageIds =
    {
            R.drawable.button1,
            R.drawable.button2,
            R.drawable.button3
    };
  /*构造方法*/
    public GanaSkyImageAdapter(Context c)
```



```
        {
            mContext = c;
        }

        // 获取图片的个数
        @Override
        public int getCount()
        {
            return mImageIds.length;
        }

        // 获取图片在库中的位置
        @Override
        public Object getItem(int position)
        {
            return position;
        }
        // 获取图片 ID
        @Override
        public long getItemId(int position)
        {
            return position;
        }
        @Override
        public View getView(int position, View convertView, ViewGroup parent)
        {
            ImageView imageView;
            if (convertView == null)
            {
                // 给 ImageView 设置资源
                imageView = new ImageView(mContext);
                // 设置布局 图片 85×85 显示
                imageView.setLayoutParams(new GridView.LayoutParams(85, 85));
                // 设置显示比例类型
                imageView.setScaleType(ImageView.ScaleType.FIT_CENTER);
            }
            else
            {
                imageView = (ImageView) convertView;
            }
            imageView.setImageResource(mImageIds[position]);
            return imageView;
        }
    }
```

代码清单 3-19　订票系统AndroidManifest.xml实现

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.ganasky"
```

```
        android:versionCode="1"
        android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_ganaSkyMain">
        <activity android:name=".GanaSkyMain"
                  android:label="@string/app_ganaSkyMain">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- -登录页面 -->
<activity android:name="LoginApp" android:label="@string/app_nameLogin"/>
          <!-- 机票查询主界面 -->
          <activity android:name="TicketMainApp" android:label="@string/app_nameTicketMain"/>

    </application>
</manifest>
```

在接下来的章节中，不赘述订票系统 AndroidManifest.xml 的界面代码，读者只要记住：每增加一个 Activity.java 文件，都需要在 AndroidManifest.xml 配置文件中引用增加的 Activity.java 文件。

代码清单 3-20　订票系统layout下的strings.xml实现

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">手机订票</string>
    <string name="app_Iconname">手机订票</string>
    <string name="app_ganaSkyMain">欢迎使用订票系统...</string>
    <string name="app_nameLogin">登录</string>
    <string name="app_nameRegister">注册</string>
     <string name="app_nameTicketMain">机票查询</string>
     <string name="app_nameCitySelect">城市选择</string>
     <string name="app_nameSelectFlight">选择航班</string>
      <string name="app_nameDetailFlight">航班详情</string>
      <string name="app_nameOrerTicket">订票</string>
</resources>
```

在 strings.xml 中，定义了每一个界面的标题栏显示的信息，在接下来的章节中会逐渐使用到。

3.3.3　程序主界面Help实现

有过程序开发经验的人，相信大家都会在程序中设计 Help 包，用来存放一些通用性较高的类，在这些类中定义开发中使用频繁的方法。在订票系统的主界面（GanaSkyMain.java）中用到了 ConnUrlHelper.hasInternet()，这个方法的功能是判断是否连通网络。详见代码清单 3-21。



代码清单 3-21　订票系统主界面（第 3 章\GanaSkyForMain）ConnUrlHelper.java

```
package com.ganasky.Help;

import android.app.Activity;
import android.content.Context;
import android.net.ConnectivityManager;
import android.net.NetworkInfo;
/**
  * 创建一个通信帮助类，将不断往这个类中添加一些与通信有关的方法
  * @author
  *
  */
public abstract class ConnUrlHelper {
	private static String DEBUG_TAG = "ConnUrlHelper";

	public static boolean hasInternet(Activity activity) {

		ConnectivityManager manager = (ConnectivityManager) activity

		.getSystemService(Context.CONNECTIVITY_SERVICE);

		NetworkInfo info = manager.getActiveNetworkInfo();

		if (info == null || !info.isConnected()) {

			return false;
		}
		if (info.isRoaming()) {

			return true;
		}
		return true;
	}

}
```

第 4 章　登录功能实现

大部分交易型的网站都提供了与用户进行交互的界面，如用户登录与注册功能，这些功能便于区分当前用户、潜在用户及非系统用户，便于系统的进一步操作，如交易支付时提供联系方式等用户信息，以便通过电话对用户进行营销。

本章涉及的基础知识有：文本框（TextView）、编辑框（EditText）、单选按钮组（RadioGroup）及按钮（Button）控件的使用。

本章涉及的知识重点有：

1）画面使用到 Android 布局。如何按实际画面需要正确地在布局中显示各种组件，本书讲解了六种布局形式。在 xml 布局文件中，采用何种布局更合理，是 Android 学习的重点。

2）登录中涉及用户名的保存。如何保存用户名，可以使用 SharedPreferences 对少量信息进行保存。

3）如何解析后台传过来的数据。在 Android 开发中，我们常把数据信息组装成 XML 格式文件，如何正确解析 XML 格式的数据量信息，也是 Android 学习的重点。

4）如何与后台进行通信。本章采用了多种通信方式，如 HttpURLConnection、HttpClient 及 Socket 方式。

4.1　基础控件讲解

在登录功能中主要涉及文本框（TextView）、编辑框（EditText）、单选按钮组（RadioGroup）及按钮（Button）控件等基础控件，这些控件在任何项目中都属于使用频率最高的控件。

4.1.1　文本框

文本框（TextView）是静态显示文本标签的控件，不与用户进行交互。常用 xml 属性见表 4-1 所示。

表 4-1　TextView 文本框常用 xml 属性

TextView 常用 xml 属性	功 能 说 明
android:text= "用户名："	设置文本内容
android:textColor = "#ff8c00"	设置文本字体颜色
android:textStyle="bold"	设置文本字体样式，如 bold， italic， bolditalic
android:textSize="20dip"	设置文本字体大小



（续）

TextView 常用 xml 属性	功 能 说 明
android:layout_gravity="center_vertical"	设置 TextView 控件在界面上显示的位置。默认 top，这里居中显示
android:gravity="center_horizontal"	设置 TextView 上的文字在控件上的位置。默认左对齐，这里居中显示
android:drawableBottom="@drawable/icon"	在 text(文本内容)的下方输出一个 drawable，如图片
android:drawableLeft="@drawable/icon"	在 text 的左边放置图片
android:drawablePadding="10dip"	设置 text 与 drawable 的间隔，与 drawableLeft、drawableRight、drawableTop、drawableBottom 一起使用，可设置为负数，单独使用没有效果
android: editable="true"	设置文本内容是否可编辑
android:scrollbars="vertical"	设置垂直滚动条
android:singleLine="false"	设置是否单行显示文本，一旦设置为 true，则文字不会自动换行，只能显示单行文本，false 则可以多行显示文本
android:maxLines="15"	设置文本内容最多不超过几行，在这里设置 15 行
android:autoLink=" all "	设置是否自动加入 URL 链接地址。单击网址时，跳向该网址链接页面，可选值有:none/web/email/phone/map/all

下面通过一个示例讲述 TextView 静态文本框的用法，该示例的主要功能是：提供两种方式显示文本信息，第一行普通方式显示文本内容，设置文本的背景色、字体颜色、字体大小等；第二行是提供超链接方式显示文本内容。代码运行结果如图 4-1 所示。



图 4-1　TextView 文本框示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-1 所示。

代码清单 4-1　TextView文本框示例（第 4 章\Demo_04_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
      <TextView
     android:id="@+id/textview1"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"
     />

     <TextView
     android:id="@+id/textview2"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:autoLink="all"
     />
</LinearLayout>
```



其次，介绍 Activity java 代码如何配合 xml 布局实现这一效果，如代码清单 4-2 所示。

代码清单 4-2 TextView文本框示例（第 3 章\Demo_04_01）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.text.Html;
import android.widget.TextView;

/**
 * TextView 文本框的使用
 * @author
 */
public class MainActivity extends Activity
{
	/* 声明 TextView 对象 */
	private TextView textview1;
	private TextView textview2;

	/** 当 activity 首次创建时响应调用 */
	@Override
	public void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		/* 获得 TextView 对象 */
		textview1 = (TextView)this.findViewById(R.id.textview1);
		textview2=(TextView)this.findViewById(R.id.textview2);

		String string = "Demo_04_01 文本框的使用！";
		/* 设置文本的颜色 */
		textview1.setTextColor(Color.WHITE);
		/* 设置字体大小 */
		textview1.setTextSize(20);
		/* 设置文字背景 */
		textview1.setBackgroundColor(Color.BLUE);
		/* 设置 TextView 显示的文字 */
		textview1.setText(string);

		/* 设置文本的颜色 */
		textview2.setTextColor(Color.WHITE);
		/* 设置字体大小 */
		textview2.setTextSize(20);
		/* 设置文字背景 */
		textview2.setBackgroundColor(Color.BLACK);
		/* 设置 TextView 的提示文字 */
		textview2.setText(Html.fromHtml("<b>textview2:</b>   "
				+ "<a href=\"http://www.google.com\">google 连接地址</a> " ));
```



```
        }
    }
```

4.1.2　文本编辑框

文本编辑框（EditText）是动态显示文本的组件，可以与用户进行交互。常用的 xml 属性如表 4-2 所示。

表 4-2　文本编辑框 EditText 常用 xml 属性

文本编辑框 EditText 常用 xml 属性	功 能 说 明
android:textColor = "#ff8c00"	设置文本字体颜色
android:textStyle="bold"	设置文本字体样式，如 bold， italic， bolditalic
android:textSize="20dip"	设置文本字体大小
android:layout_gravity="center_vertical"	设置控件在界面上显示的位置。默认 top，这里居中显示
android:gravity="center_horizontal"	设置控件上的文字在控件上的位置。默认左对齐，这里居中显示
android:hint="请输入数字！"	设置显示在空间上的提示信息
android:numeric="integer"	设置只能输入整数，如果是小数则是：decimal
android:password="true"	设置只能输入密码
android:phoneNumber="true"	输入只能输入电话号码
android:editable="true"	设置文本内容是否可编辑
android:scrollbars="vertical"	设置垂直滚动条
android:singleLine="false"	设置是否单行显示文本，一旦设置为 true，则文字不会自动换行，只能显示单行文本，false 则可以多行显示文本
android:maxLines="15"	设置文本内容最多不超过几行，在这里设置 15 行
android:autoLink="all"	设置是否自动加入 URL 链接地址。单击网址时，跳向该网址链接页面，可选值有 :none/web/email/phone/map/all

下面通过一个示例讲述 EditText 动态编辑框的用法，该示例的主要功能是：包括两种方式显示编辑框的文字内容，普通方式显示 EditText 文字及提示方式显示 EditText 文字，通过设置不同的文本样式，让读者能够更清晰看出两者的区别。代码运行结果如图 4-2 所示。



图 4-2　文本编辑框 EditText 运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-3 所示。

代码清单 4-3　文本编辑框EditText 示例（第 4 章\Demo_04_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
```

```
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <TextView
            android:id="@+id/TextView01"
            android:layout_width="fill_parent"
             android:layout_height="wrap_content" />
        <EditText android:id="@+id/EditText01"
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:textSize="18sp"
             android:layout_x="29px"
             android:layout_y="33px"    />
        <EditText android:id="@+id/EditText02"
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:textSize="18sp"
             android:layout_x="29px"
             android:layout_y="33px" />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 4-4 所示。

代码清单 4-4　文本编辑框EditText示例（第 4 章\Demo_04_02）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.widget.EditText;
import android.widget.TextView;

/**
 * Demo_04_02,EditText 的使用
 * @author
 */
public class MainActivity extends Activity {
        /*  声明 TextView 对象  */
        private TextView m_TextView;
        /*  声明 EditText 对象  */
        private EditText m_EditText1;
        private EditText m_EditText2;

        /**  当 activity 首次创建时响应调用  */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                /*  获得 TextView 对象  */
```



```
        m_TextView = (TextView) findViewById(R.id.TextView01);
        /* 获得 EditText 对象 */
        m_EditText1 = (EditText) findViewById(R.id.EditText01);
        m_EditText2=(EditText) findViewById(R.id.EditText02);

        /* 设置 EditTextw 显示的文字 */
        m_EditText1.setText("请输入用户名");

        /* 提示框方式显示 EditTextw 的文字 */
        m_EditText2.setHint("请输入用户名");

        /* 设置字体大小 */
        m_EditText1.setTextSize(20);
        m_EditText2.setTextSize(20);

        /* 设置字体颜色 */
        m_EditText1.setTextColor(Color.RED);
        m_EditText2.setTextColor(Color.GRAY);

        /* 设置 EditText 事件监听 */
        m_EditText1.setOnKeyListener(m_EditText1OnKey);

    }
    /**
     * 实现事件监听
     */
    EditText.OnKeyListener    m_EditText1OnKey=new EditText.OnKeyListener(){
        @Override
        public boolean onKey(View v, int keyCode, KeyEvent event) {
            // TODO Auto-generated method stub
            // 得到文字，将其显示到 TextView 中
            m_TextView
                    .setText("文本框中输入的内容是：" + m_EditText1.getText().toString());
            return false;
        }

    };
}
```

4.1.3　单选按钮

单选按钮（RadioButton）可以实现选择某一项子项的功能，可以与用户进行交互，一般与 RadioGroup 配合使用。其重要的方法是 OnCheckedChangeed，用户可以监听 OnCheckedChangeListener，实现 onCheckedChanged 方法，捕获到某一子项的 ID，从而实现切换选择子项的功能。下面通过一个示例讲述 RadioButton 的用法，主要功能是单项选择某一项并获取该项的值，代码运行结果如图 4-3 所示。



图 4-3　单选按钮组示例代码运行结果图

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-5 所示。

代码清单 4-5　单选按钮组示例（第 4 章\Demo_04_03）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
      <TextView
      android:id="@+id/TextView01"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"
     />
   <RadioGroup
     android:id="@+id/RadioGroup01"
     android:layout_width="wrap_content"
     android:layout_height="wrap_content"
     android:orientation="vertical"
     android:layout_x="3px"
     android:layout_y="54px"
     >
     <RadioButton
        android:id="@+id/RadioButton1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/RadioButton1"
     />
     <RadioButton
        android:id="@+id/RadioButton2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/RadioButton2"
     />
     <RadioButton
        android:id="@+id/RadioButton3"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/RadioButton3"
     />
     <RadioButton
        android:id="@+id/RadioButton4"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="@string/RadioButton4"
     />
     </RadioGroup>
</LinearLayout>
```



其次，介绍 strings.xml 如何配合 xml 布局实现这一效果，如代码清单 4-6 所示。

代码清单 4-6　单选按钮组示例（第 4 章\Demo_04_03）strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string name="hello">目前最火的手机开发是？</string>
    <string name="app_name">Demo_04_03 单选按钮组的使用</string>
    <string name="RadioButton1">Android 开发</string>
    <string name="RadioButton2">WP7 开发</string>
    <string name="RadioButton3">iphone  开发</string>
    <string name="RadioButton4">Flex 开发</string>
</resources>
```

再者，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 4-7 所示。

代码清单 4-7　单选按钮组示例（第 4 章\Demo_04_03）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.widget.RadioButton;
import android.widget.RadioGroup;
import android.widget.TextView;
import android.widget.Toast;

/**
 * Demo_04_03,RadioGroup 的使用
 * @author
 */
public class MainActivity extends Activity {
    /**
     *  创建 TextView 对象  创建 RadioGroup 对象  创建 4 个 RadioButton 对象
     */
    TextView m_TextView;
    RadioGroup m_RadioGroup;
    RadioButton m_Radio1, m_Radio2, m_Radio3, m_Radio4;

    /**  当 activity 首次创建时响应调用  */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /**
         *  获得 TextView 对象  获得 RadioGroup 对象  获得 4 个 RadioButton 对象
         */
        m_TextView = (TextView) findViewById(R.id.TextView01);
        m_RadioGroup = (RadioGroup) findViewById(R.id.RadioGroup01);
        m_Radio1 = (RadioButton) findViewById(R.id.RadioButton1);
        m_Radio2 = (RadioButton) findViewById(R.id.RadioButton2);
```

```
        m_Radio3 = (RadioButton) findViewById(R.id.RadioButton3);
        m_Radio4 = (RadioButton) findViewById(R.id.RadioButton4);

        /* 设置事件监听 */
        m_RadioGroup.setOnCheckedChangeListener(m_RadioGroupChoice);
    }
    /**
     * 实现事件监听
     */
    RadioGroup.OnCheckedChangeListener    m_RadioGroupChoice=
        new RadioGroup.OnCheckedChangeListener(){

        @Override
        public void onCheckedChanged(RadioGroup group, int checkedId) {
            // TODO Auto-generated method stub
            if (checkedId == m_Radio1.getId()) {
                DisplayToast("您选择了：" + m_Radio1.getText());
            } else if (checkedId == m_Radio2.getId()) {
                DisplayToast("您选择了：" + m_Radio2.getText());
            }else if (checkedId == m_Radio3.getId()) {
                DisplayToast("您选择了：" + m_Radio3.getText());
            }else if (checkedId == m_Radio4.getId()) {
                DisplayToast("您选择了：" + m_Radio4.getText());
            }

        }
    };

    /* 显示 Toast */
    public void DisplayToast(String str) {
        Toast toast = Toast.makeText(this, str, Toast.LENGTH_LONG);
        // 设置 toast 显示的位置
        toast.setGravity(Gravity.TOP, 0, 200);
        // 显示该 toast
        toast.show();
    }
}
```

4.1.4 按钮控件

按钮（Button）是 Android 常用控件，在前面的示例中也曾多次涉及。它是一个按钮控件，因此具备事件处理的功能，当用户单击某一按钮的时候，会触发相应事件，如单击、双击及触摸屏等事件，Button 按钮常见事件在第 3 章 Android 事件处理机制中涉及过，详见表 3-4。



图 4-4 Button 按钮实现拖拽示例代码运行结果

下面通过一个示例讲述 Button 的用法，该示例实现拖拽按钮的功能，代码运行结果如图 4-4 所示。

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-8 所示。



代码清单 4-8　Button按钮实现拖拽示例（第 4 章\Demo_04_04）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">

            <TextView
            android:id="@+id/TextView01"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:text="@string/hello"
           />
            <Button
                 android:id="@+id/btn_Drag"
                 android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:text="拖拽我吧！！"
            />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 4-9 所示。

代码清单 4-9　Button按钮实现拖拽示例（第 4 章\Demo_04_04）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;

/**
 * Button 按钮实现拖拽
 * @author
 */
public class MainActivity extends Activity {
      /**  当 activity 首次创建时响应调用  */
      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
            /*  获得 Button 对象  */
            final Button btn = (Button) findViewById(R.id.btn_Drag);
            /*  设置事件监听  */
            btn.setOnTouchListener(DragButton);
      }
```




```
    /**
     * 实现事件监听
     */
    Button.OnTouchListener DragButton = new Button.OnTouchListener() {
        int[] temp = new int[] { 0, 0 };

        @Override
        public boolean onTouch(View v, MotionEvent event) {
            /* 获得动作类型 */
            int eventaction = event.getAction();
            /* 写日志 */
            Log.i("&&&", "onTouchEvent:" + eventaction);
            /* 获得坐标点 */
            int x = (int) event.getRawX();
            int y = (int) event.getRawY();

            /* 判断动作类型，并进行相应操作 */
            switch (eventaction) {
            /* 按下时改变物体（按钮）的坐标 */
            case MotionEvent.ACTION_DOWN:
                temp[0] = (int) event.getX();
                temp[1] = y - v.getTop();
                break;
            /* 移动时改变物体（按钮）的坐标 */
            case MotionEvent.ACTION_MOVE:
                v.layout(x - temp[0], y - temp[1], x + v.getWidth() - temp[0],
                        y - temp[1] + v.getHeight());
                v.postInvalidate();
                break;
            /* 按上时改变物体（按钮）的坐标 */
            case MotionEvent.ACTION_UP:
                break;
            }
            return false;
        }

    };
}
```

4.2 重点剖析

本节介绍功能界面实现中需要重点掌握的 Android 编程方法与编程思路，主要介绍 Android 长度单位、Android 各种布局的用法、SharedPreferences 的使用、Android 解析 XML 的方法、Android 客户端与后台通信等内容。

4.2.1 Android长度单位

手机屏幕具有不同的尺寸大小，如何编写一次应用程序就可以在不同的手机屏幕及不同手机分辨率中正常且美观显示出来呢？那就需要考虑长度单位的使用。长度单位分相对单位



和绝对单位两种。相对单位主要包括：px、sp、dp、dip，绝对单位包括：pt、in、mm。一般用相对单位，而不用绝对单位。 Android 长度单位的使用几乎在每一个示例中都涉及，在这里主要是做一次总结，阐述长度单位的具体含义，不举例说明其具体用法。

（1）Android 长度单位含义

Android 长度单位具体含义如表 4-3 所示。

表 4-3　Android 长度单位含义详解

单　位	含　义
px（像素）	屏幕上的点
dp（与密度无关的像素）	一种基于屏幕密度的抽象单位。在每英寸 160 点的显示器上，1dp = 1px
dip	与 dp 相同，多用于 Google 示例中
sp（与刻度无关的像素）	与 dp 类似，但是可以根据用户的字体大小首选项进行缩放
in（英寸）	长度单位
mm（毫米）	长度单位
pt（磅）	1/72 英寸

（2）Android 长度单位应用总结

在开发过程中，尽量使用 dp 作为空间大小单位，sp 作为文字相关大小单位。

1）设置字体的大小一般使用 sp，用此单位的字体能够根据用户设置字体的大小而自动缩放。

2）设置表示长度、高度等属性时可以使用 dp 或 sp。

3）空间等相对距离一般使用 dp（dip）随着密度变化，对应的像素数量也变化，但并没有直接的相对比例的变化。

4）px 与实际像素有关，还与密度有关。dp 和 sp 和实际像素没有关系，对于一定分辨率但不同密度的屏幕，px 单位的应用可能会导致长度的相对比例的变化。

（3）密度与分辨率

密度值表示每英寸有多少个显示点，与分辨率是两个概念。

各自 Android 手机屏幕密度（density）标准是：HVGA 屏的 density=160；QVGA 屏 density=120；WVGA 屏 density=240；WQVGA 屏 density=120。

具体的应用运算关系：假设分辨率是 x*y，密度为 d，屏幕实际大小为 a*b 那么关系为 x*y = d * a * b（约等于）。

不同 density 下屏幕分辨率信息不同。density=240 时 ，屏幕实际分辨率为 480dip*800dip；density=120 时，屏幕实际分辨率为 240px*400px （两个点对应一个分辨率）。

（4）密度与分辨率对比总结

1）在相同密度（即同一实体屏幕）不同分辨率的情况下，与实体密度无关的相对单位（sp 和 dp）显示正常。

2）在相同分辨率不同密度的情况下，都用的标准密度，所以分析的意义不是很大。

（5）Android 控件宽度和长度设置

一般情况下，无须为文本控件或者编辑控件的宽和高设置具体的值，可以使用相对设置，便于实现自适应。如 android:layout_width="fill_parent"，定义当前视图在屏幕上的可以使用的宽度，fill_parent 填充整个屏幕；android:layout_width="wrap_content"，则表示的是控

件的宽度是随着控件内容的多少自动调整宽度。在实际项目开发中，一般控件的长度可以这样设置：android:layout_width="fill_parent"，而宽度则是：android:layout_height="wrap_content"，也可以根据实际需要进行设置。

4.2.2 Android布局的用法

在 Android 开发中，会经常接触到布局文件 Layout.xml，Layout 对于快速搭建界面和提高界面在不同分辨率屏幕上的适应性具有重要的作用。Android Layout 布局的方式有 LinearLayout（线性布局，包括垂直线性布局和水平线性布局）、AbsoluteLayout（绝对布局）、RelativeLayout（相对布局）、TableLayout（表单布局）四种。TabWidget（切换卡）常与这四种布局一起配合使用，因此放在此处一并讲解。下面介绍放入 Layout 中进行排列显示的子组件（View）具有哪些常见的 XML 属性，见表 4-4。

表 4-4 Layout 中的子组件常见的 XML 属性

布 局 分 类	属性及功能介绍
四种 Layout 中的 Item 所共有的 XML 属性	1）layout_width 用来确定放入 Layout 中的子控件的宽度 2）layout_height 用来确定放入 Layout 中的子控件的高度 注：1）和 2）它们的可能取值为 fill_parent，wrap_content 或者固定的像素值 3）layout_marginLeft <-->paddingLeft （左边界） 4）layout_marginTop <-->paddingTop （上边界） 5）layout_marginRight <-->paddingRight （右边界） 6）layout_marginBottom <-->paddingBottom （下边界） 7）layout_margin <-->padding （边界，包括上下左右） 注: 3）、4）、5）、6）、7）表示的是：放入 Layout 中的子控件与 Layout 的边界或者其他子控件之间能够相距一段距离。进一步解析那就是，“<-->”左边的 xml 属性表示的是：放入布局中的子控件与布局的边界之间的相距距离，右边的 xml 属性表示的是：放入布局中的子控件之间的边界的相距距离。 8） layout_gravity:用来确定子控件在 Layout 中的停靠位置 注：FrameLayout 是最简单的布局，只有上述这几个属性。
LinearLayout 独有的 xml 属性	layout_weight: 用于在 LinearLayout 中把所有子控件排列之后的剩余空间按照它们的 layout_weight 分配给各个拥有此属性的子控件
RelativeLayou 独有的 xml 属性	第一类:属性值为 true 或 false android:layout_centerHrizontal 水平居中 android:layout_centerVertical 垂直居中 android:layout_centerInparent 相对于父元素完全居中 android:layout_alignParentBottom 贴紧父元素的下边缘 android:layout_alignParentLeft 贴紧父元素的左边缘 android:layout_alignParentRight 贴紧父元素的右边缘 android:layout_alignParentTop 贴紧父元素的上边缘 android:layout_alignWithParentIfMissing 如果参照（兄弟）元素找不到的话就以父元素作参照物 第二类：属性值必须为 id 的引用名” @+id/id-name” android:layout_below 在某元素的下方 android:layout_above 在某元素的上方 android:layout_toLeftOf 在某元素的左边 android:layout_toRightOf 在某元素的右边 android:layout_alignTop 本元素的上边缘和某元素的上边缘对齐 android:layout_alignLeft 本元素的左边缘和某元素的左边缘对齐 android:layout_alignBottom 本元素的下边缘和某元素的下边缘对齐 android:layout_alignRight 本元素的右边缘和某元素的右边缘对齐 第三类：属性值为具体的像素值，如 30dip，40px android:layout_marginBottom 离其他元素底边缘的距离 android:layout_marginLeft 离其他元素左边缘的距离 android:layout_marginRight 离其他元素右边缘的距离 android:layout_marginTop 离其他元素上边缘的距离



（续）

布 局 分 类	属性及功能介绍
TableLayout 独有的属性	1）android:layout_span:设置合并几个单元格 2）android:layout_column:设置 index 值并实现跳开某些单元格 3）android:collapseColumns：以第 0 行为序，隐藏指定的列。如把 android:collapseColumns=0,2 意思是把第 0 和第 2 列隐藏 4）android:shrinkColumns：以第 0 行为序，自动延伸指定的列填充可用部分。当 LayoutRow 里面的控件还没有布满布局时，shrinkColumns 不起作用,设置了 shrinkColumns=0,1,2，布局完全没有改变，因为 LayoutRow 里面还剩足够的空间。当 LayoutRow 布满控件时,设置了 shrinkColumns=2，则控件自动向垂直方向填充空间 5）android:stretchColumns：以第 0 行为序，尽量把指定的列填充空白部分。设置 stretchColumns=1，第 1 列被尽量填充

需要注意的是：根据 Android 开发文档，Android 会对 Layout 及其子控件嵌套组成的这棵树进行两次遍历：一次是 measure 调用，用来确定 Layout 及其子控件的大小；一次是 layout 调用，用来确定 Layout 或者子控件的位置。下面通过示例分别讲述这几种 Layout 布局的用法。

1. 垂直线性布局

垂直线性布局（LinearLayout），使用 LinearLayout 标签，在标签属性中加 android:orientation="vertical"，所有的子项就会按照“线性垂直”方式排序。LinearLayout 中有一个重要的属性：android:layout_weight="1"，这个 weight 在垂直布局时，代表行距。在此布局下，每一行只能放一个控件。

下面的示例讲述 Android 垂直布局的使用方法，示例的功能很简单，垂直排列显示多个文本框，示例代码运行结果如图 4-5 所示。

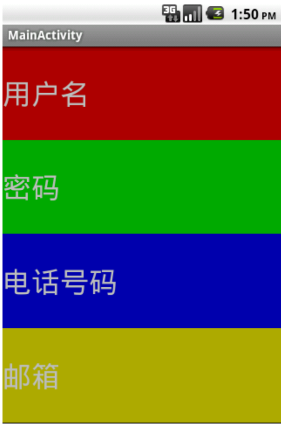


图 4-5　垂直线性布局显示文本信息

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-10 所示。

代码清单 4-10　垂直线性布局示例（第 4 章\Demo_04_05）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
      <TextView
          android:text="用户名"
          android:gravity="center_vertical"
          android:textSize="15pt"
          android:background="#aa0000"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>
```

```
    <TextView
        android:text="密码"
        android:textSize="15pt"
        android:gravity="center_vertical"
        android:background="#00aa00"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"/>

    <TextView
        android:text="电话号码"
        android:textSize="15pt"
        android:gravity="center_vertical"
        android:background="#0000aa"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"/>
    <TextView
        android:text="邮箱"
        android:textSize="15pt"
        android:gravity="center_vertical"
        android:background="#aaaa00"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1"/>

</LinearLayout>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现这一效果，如代码清单 4-11 所示。

代码清单 4-11 垂直线性布局示例（第 4 章\Demo_04_05）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
/**
 * Demo_04_05 垂直布局的使用
 * @author
 */
public class MainActivity extends Activity
{
    /** 当 activity 首次创建时响应调用 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```



2. 水平线性布局

水平线性布局（LinearLayout），使用的也是LinearLayout 标签，在标签属性中加 android:orientation="horizontal"，所有的子项就会按照“线性水平”排序。linearLayout 中也有 android:layout_weight="1"属性，这个 weight 在水平布局中时，代表列宽，weight 值越大表示列宽越大。在此布局下，每一列只能放一个控件。

下面的示例讲述 Android 水平布局的使用方式，示例的功能很简单，水平排列显示多个文本框，示例代码运行结果如图 4-6 所示。



图 4-6　水平线性布局显示文本信息

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-12 所示。

代码清单 4-12　水平线性布局使用示例（第 4 章\Demo_04_06）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="horizontal"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
      <TextView
          android:text="第一格"
          android:gravity="center_vertical"
          android:textSize="15pt"
          android:background="#aa0000"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>

      <TextView
          android:text="第二格"
          android:textSize="15pt"
          android:gravity="center_vertical"
          android:background="#00aa00"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>

      <TextView
          android:text="第三格"
          android:textSize="15pt"
          android:gravity="center_vertical"
          android:background="#0000aa"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1"/>

      <TextView
```

```
            android:text="第四格"
            android:textSize="15pt"
            android:gravity="center_vertical"
            android:background="#aaaa00"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1"/>

</LinearLayout>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现这一效果，如代码清单 4-13 所示。

代码清单 4-13 水平线性布局使用示例（第 4 章\Demo_04_06）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
/**
 * Demo_04_06 水平线性布局的使用
 * @author
 */
public class MainActivity extends Activity
{
    /** 当 activity 首次创建时响应调用 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

3. 绝对布局

绝对布局（AbsoluteLayout）是指各个元素在界面上的位置是基于坐标，Android 手机的坐标系是以左上定点为原点坐标（0,0），向右为 X 轴正方向，向下为 Y 轴正方向。绝对布局犹如 HTML 页面布局中的 div 指定了 absolute 属性，用 X、Y 坐标来指定元素的位置，如 android:layout_x="20px" 、android:layout_y="12px"。这种布局方式在垂直切换时，往往会出问题，而且多个元素的时候，计算坐标位置比较麻烦。

下面的示例讲述 Android 绝对布局的使用方式，示例代码运行结果如图 4-7 所示。



图 4-7 绝对布局显示文本信息

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-14 所示。

代码清单 4-14 绝对布局使用示例（第 4 章\Demo_04_07）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```



```
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView
         android:text="用户名:"
         android:gravity="center_vertical"
         android:textSize="15pt"
         android:background="#aa0000"
         android:layout_width="120dip"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:layout_x="5px"
         android:layout_y="5px"/>

     <EditText
          android:id="@+id/et_UserName"
          android:layout_width="150dip"
          android:layout_height="wrap_content"
          android:textSize="11sp"
          android:hint="请输入用户名"
          android:layout_x="125px"
         android:layout_y="5px"/>

     <TextView
         android:text="密码    :"
         android:textSize="15pt"
         android:gravity="center_vertical"
         android:background="#00aa00"
         android:layout_width="120dip"
         android:layout_height="wrap_content"
         android:layout_weight="1"
         android:layout_x="5px"
         android:layout_y="150px"/>

      <EditText
           android:id="@+id/et_UserPwd"
           android:layout_width="150dip"
           android:layout_height="wrap_content"
           android:textSize="11sp"
           android:password="true"
           android:hint="请输入密码"
           android:layout_x="125px"
          android:layout_y="150px"/>
</AbsoluteLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 4-15 所示。

代码清单 4-15　绝对布局使用示例（第 4 章\Demo_04_07）MainActivity.java

```
package com.chen.android;
```




```
import android.app.Activity;
import android.os.Bundle;
/**
 * Demo_04_07 绝对布局的使用
 * @author
 */
public class MainActivity extends Activity
{
	/** 当 activity 首次创建时响应调用 */
	@Override
	public void onCreate(Bundle savedInstanceState)
	{
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
	}
}
```

4. 相对布局

相对布局（RelativeLayout）是选择某一个元素作为参照物，来定位具体元素的布局方式。主要的 xml 属性见表 4-4 所示，下面的示例讲述 Android 相对布局的使用方法，示例代码运行结果如图 4-8 所示。



图 4-8　相对布局显示控件

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-16 所示。

代码清单 4-16　相对布局使用示例（第 4 章\Demo_04_08）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
        android:id="@+id/label1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="请输入用户名:"/>

    <EditText
        android:id="@+id/entry1"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:background="@android:drawable/editbox_background"
        android:layout_below="@id/label1"/>
```



```
        <TextView
            android:id="@+id/label2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="请输入密码:"
            android:layout_below="@id/entry1"/>

        <EditText
            android:id="@+id/entry2"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:background="@android:drawable/editbox_background"
            android:layout_below="@id/label2"/>

        <Button
            android:id="@+id/ok"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_below="@id/entry2"
            android:layout_alignParentRight="true"
            android:layout_marginLeft="10dip"
            android:text="确定" />
        <Button
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_toLeftOf="@id/ok"
            android:layout_alignTop="@id/ok"
            android:text="取消" />
</RelativeLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 4-17 所示。

代码清单 4-17　相对布局使用示例（第 4 章\Demo_04_08）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
/**
 * Demo_04_08 相对布局的使用
 * @author
 */
public class MainActivity extends Activity {
    /** 当 activity 首次创建时响应调用 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```



5. 表单布局

表格布局（TableLayout）类似 html 中的 table。每一个 TableLayout 里面有表格行 TableRow，它是一个横向的水平（horizontal）的 LinearLayout。TableRow 里面可以具体定义每一个元素。设定表格元素的对齐方式，可以使用 android:gravity=" "。但与我们平时所见的网页 Table 有所不同，TableLayout 没有边框，它是由多个 TableRow 对象组成，每个 TableRow 可以有 0 个或多个单元格，每个单元格就是一个 View。这些 TableRow 里面的单元格不能设置 layout_ width，宽度默认是 fill_parent，只有高度 layout_ height 可以自定义，默认是 wrap_content。单元格可以为空，并且可以通过 android:layout_column 设置 index 值并实现跳开某些单元格。在 TableRow 之间，可以添加 View、设置 layout_height 以及背景色，就可以实现一条间隔线。还可以通过 android:layout_span 设置合并几个单元格。通过下面的示例讲述 Android 表单布局相关属性的具体使用方法，示例代码运行结果如图 4-9 所示。



图 4-9　表单布局

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-18 所示。

代码清单 4-18　表单布局使用示例（第 4 章\Demo_04_09）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TableLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">

     <TableRow>
          <TextView
               android:text="column1"
               android:padding="3dip" />
          <TextView
               android:text="column2"
               android:padding="3dip" />
          <TextView
              android:text="column3"
              android:padding="3dip" />
          <TextView
              android:text="column4"
              android:padding="3dip" />
     </TableRow>

     <TableRow>
          <TextView
              android:text="column11"
              android:visibility="invisible" />
```



```
            <TextView
                android:text="左边的 invisible"
                android:gravity="right"
                 android:padding="3dip" />
            <Button
                 android:id="@+id/go"
                 android:width="60dip"
                 android:text="go"
                 android:padding="3dip" />
            <Button
                 android:id="@+id/cancel"
                 android:width="60dip"
                 android:text="cancel"
                 android:padding="3dip" />
        </TableRow>

        <View
               android:layout_height="2dip"
               android:background="#F00" />

        <TableRow>
            <TextView
                android:text="右边的单元格" />
            <TextView
                android:layout_column="2"
                android:text="跳开的单元格"
                 android:padding="3dip" />
        </TableRow>

        <TableRow>
            <TextView
                 android:text="合并 4 个单元格"
                 android:layout_span="4"
                 android:gravity="center_horizontal"
                 android:background="#FFC0C0C0"
                 android:textColor="#f00"
                 android:padding="3dip" />
        </TableRow>

    </TableLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 4-19 所示。

代码清单 4-19　表单布局使用示例（第 4 章\Demo_04_09）MainActivity.java

```
    package com.chen.android;

    import android.app.Activity;
    import android.os.Bundle;
    /**
      * Demo_04_09 绝对布局的使用
```

```
 * @author
 */
public class MainActivity extends Activity {
    /** 当 activity 首次创建时响应调用 */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    }
}
```

6. 切换卡

（Tab）切换卡（TabWidget）的显示界面类似于手机电话薄的界面，通过多个标签来切换显示不同的屏幕内容。TabWidget 继承自 TabActivity 类，并实现 setOnTabChangedListener 的 onTabChanged 方法来监听 Tab 的改变。要实现这一效果，首先要知道 TabHost，它是一个用来存放多个 Tab 标签的容器，每一个 Tab 都可以对应自己的布局。比如，电话本中的 Tab 布局就是一个线性布局，要使用 TabHost，首先要通过 getTabHost 方法获取 TabHost 的对象，然后通过 addTab 方法来向 TabHost 中添加 Tab，当然每个 Tab 在切换时都会产生一个事件（onTabChanged 事件），要捕捉这个事件，需要设置 TabActivity 的事件监听 setOnTabChangedListener。

下面通过一个示例讲述 Android 切换卡的使用方法，示例代码运行结果如图 4-10 所示。



图 4-10　切换卡示例运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-20 所示。

代码清单 4-20　切换卡使用示例（第 4 章\Demo_04_10）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     xmlns:android="http://schemas.android.com/apk/res/android">

     <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
        <TabWidget android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"/>

        <FrameLayout
          android:id="@android:id/tabcontent"
          android:layout_width="fill_parent"
```



```
            android:layout_height="fill_parent"
            android:padding="5dp" />

    </LinearLayout>
</TabHost>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 4-21 至代码清单 4-23 所示。

代码清单 4-21　切换卡使用示例（第 4 章\Demo_04_10）MainActivity.java

```
package com.chen.android;

import android.app.TabActivity;
import android.content.Intent;
import android.content.res.Resources;
import android.os.Bundle;
import android.widget.TabHost;
import android.widget.TabHost.TabSpec;
 /**
  * Demo_04_10 Android 选项卡（TabWidget）例子
  * @author
  */
public class MainActivity extends TabActivity {

    /**  当 activity 首次创建时响应调用  */
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /*  调用 layout 中定义的 main.xml */
        setContentView(R.layout.main);
        /*获取图片资源  */
        Resources res = getResources();
        /*获取 TabHost */
        TabHost tabHost = getTabHost();
        TabSpec spec;
        /*定义  Intent 页面*/
        Intent intent;

        /*第一个 TAB*/
      /*新建一个 Intent 用作 Tab1 显示的内容*/
        intent = new Intent(this,OneActivity.class);
        spec = tabHost.newTabSpec("tab1")//新建一个  Tab
        .setIndicator("Tab1",
         res.getDrawable(android.R.drawable.ic_media_play))//设置名称以及图标
        .setContent(intent);//设置显示的 intent，这里的参数也可以是 R.id.xxx
        tabHost.addTab(spec);//添加进 tabHost

        /*第二个 TAB*/
        intent = new Intent(this,TwoActivity.class);//第二个 Intent 用作 Tab1 显示的内容
        spec = tabHost.newTabSpec("tab2")//新建一个  Tab
```




```
                .setIndicator("Tab2",
                 res.getDrawable(android.R.drawable.ic_menu_camera))//设置名称以及图标
                .setContent(intent);//设置显示的 intent，这里的参数也可以是 R.id.xxx
                tabHost.addTab(spec);//添加进 tabHost

                /*当前显示第一个 TAB*/
                tabHost.setCurrentTab(1);
            }

    }
```

代码清单 4-22　切换卡使用示例（第 4 章\Demo_04_10）OneActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class OneActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView textview = new TextView(this);
        textview.setText("这是选项卡一");
        setContentView(textview);
    }
}
```

代码清单 4-23　切换卡使用示例（第 4 章\Demo_04_10）TwoActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.TextView;

public class TwoActivity extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        TextView textview = new TextView(this);
        textview.setText("这是选项卡二");
        setContentView(textview);


    }
}
```

再者，在 AndroidManifest.xml 中，需要添加这两个选项卡，如代码清单 4-24 所示。

代码清单 4-24　切换卡使用示例（第 4 章\Demo_04_10）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon"
                        android:label="@string/app_name">
        <activity android:name=".MainActivity"
                        android:theme="@android:style/Theme.NoTitleBar"
                        android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="OneActivity"></activity>
        <activity android:name="TwoActivity"></activity>

    </application>
</manifest>
```

4.2.3　SharedPreferences的使用

SharedPreferences 是 Android 平台上一个轻量级的存储类，它是以一种简单、透明的方式来保存一些用户个性化设置的字体、颜色、位置等常用配置。它主要用于系统配置信息的保存，如一般的应用程序都会提供“设置”或者“首选项”这样的界面；用户登录时输入了用户名、密码，下次登录时保留这一次的输入信息。这些设置大多是通过轻量级的存储类 Preferences 来完成保存功能。

SharedPreferences 主要是以键值对来存储应用程序的配置信息，它只能存储基本数据类型。一个程序的配置文件仅可以在本应用程序中使用，或者说只能在同一个包内使用，不能在不同的包之间使用。实际上，SharedPreferences 是采用了 XML 格式将数据存储到设备中，在 DDMS 中的 File Explorer 中的/data/data/<package name>/shares_prefs 下可查看所存储的信息。下面介绍获取 SharedPreferences 对象的两个方法，详见表 4-5、4-6 所示。

表 4-5　获取 SharedPreferences 对象的两个方法

返　回　值	函　　数	功 能 说 明
SharedPreferences	Context.getSharedPreferences (String name，int mode)	name 为本组件的配置文件名（如果想要与本应用程序的其他组件共享此配置文件，可以使用 name 来检索到这个配置文件） mode 为操作模式，默认的模式为 0 或 MODE_PRIVATE，还可以使用 MODE_WORLD_READABLE 和 MODE_WORLD_WRITEABLE，具体含义详见表 4-6
SharedPreferences	Activity.getPreferences(int mode)	配置文件仅供被调用的 Activity 使用 mode 为操作模式，默认的模式为 0 或 MODE_PRIVATE，还可以使用 MODE_WORLD_READABLE 和 MODE_WORLD_WRITEABLE

表 4-6　android 文件存储的 4 种方式

属　　性	功 能 说 明
Context.MODE_PRIVATE	默认操作模式，代表该文件是私有数据，只能被应用程序本身访问，在该模式下，写入的内容会覆盖原文件的内容
Context.MODE_APPEND	模式会检查文件是否存在，存在就往文件里追加内容，否则就创建新文件
MODE_WORLD_READABLE	表示当前文件可以被其他应用读取
MODE_WORLD_WRITEABLE	表示当前文件可以被其他应用写入

在代码中如何具体使用 SharedPreferences 呢？如果要读取配置文件信息，只需直接使用 SharedPreferences 对象的 getXXX()方法即可，而如果要写入配置信息，则必须先调用 SharedPreferences 对象的 edit()方法，使其处于可编辑状态，接着再调用 putXXX()方法写入配置信息，最后调用 commit()方法提交更改后的配置文件。需要注意的是要实现页面传值及读取文件信息的功能（即在一个页面保存，另外一个页面可以读取信息）则需使用第一种 Context.getSharedPreferences(String name，int mode)方式，且访问权限至少是可读权限（MODE_WORLD_READABLE）。下面通过一个示例讲述 SharedPreferences 具体的使用过程，示例代码运行结果如图 4-11 和图 4-12 所示。



图 4-11　SharedPreferences 保存用户信息



图 4-12　第二次启动程序，显示用户名

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-25 所示。

代码清单 4-25　SharedPreferences保存信息示例（第 4 章\Demo_04_11）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
   <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```



```
            android:text="用户名"
        />
    <EditText
        android:id="@+id/txtUserName"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:maxLength = "20"
        />
    <TextView
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="密码"
            />
    <EditText
        android:id="@+id/txtPass"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:maxLength = "20"
        android:password = "true"
        />
     <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
          >
     <Button
        android:id="@+id/btnSave"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="保存"
        android:gravity = "center"
        android:width = "80px"
        />
     </LinearLayout>
</LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现 SharedPreferences 保存信息这一功能，如代码清单 4-26 所示。

代码清单 4-26　SharedPreferences示例（第 4 章\Demo_04_11）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
```




```
import android.widget.EditText;
import android.widget.Toast;
import android.content.Context;
import android.content.SharedPreferences;
/**
 * Demo_04_11SharedPreferences 的用法
 * @author
 *
 */
public class MainActivity extends Activity {
    /**当首次创建 activity 时被调用*/
    private String saveOfUserName = null;
    public static final String PREFS_NAME = "保存用户信息";
    /* 声明 EditText 对象 */
    private EditText m_UserName;
    private EditText m_UserPwd;
    private Button   m_Save;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        /*获取 main.xml 中的对象*/
        m_UserName=(EditText) findViewById(R.id.txtUserName);
        m_UserPwd=(EditText) findViewById(R.id.txtPass);
        m_Save=(Button)findViewById(R.id.btnSave);

        /*设置事件监听*/
        m_Save.setOnTouchListener(OnTouchsaveInfo);

        /*读取用户信息*/
        readUserName();
    }
    /**
     * 保存用户信息
     */
    Button.OnTouchListener    OnTouchsaveInfo=new Button.OnTouchListener(){
        @Override
        public boolean onTouch(View v, MotionEvent event) {

            /*保存用户信息*/
            SaveUserInfo();
            DisplayToast("保存用户信息成功！");
            return false;
        }
    };

    /**
     * 读取信息
     */
    private void readUserName() {
```



```
			// 取得活动的 preferences 对象.
			SharedPreferences settings = getSharedPreferences(PREFS_NAME,
					Context.MODE_WORLD_READABLE);
			// 取得值
			saveOfUserName = settings.getString("userName", saveOfUserName);
			if(saveOfUserName!=null && saveOfUserName!="")
			{
			m_UserName.setText(saveOfUserName);
			}
		}
		/**
		 * 是否保存用户
		 */
		private void SaveUserInfo() {
			// 保存
			SharedPreferences uiState = getSharedPreferences(PREFS_NAME,
					Context.MODE_WORLD_READABLE);
			// 取得编辑对象
			SharedPreferences.Editor editor = uiState.edit();
			// 保存用户信息
				saveOfUserName=m_UserName.getText().toString();
				if (saveOfUserName != null) {
				editor.putString("userName",saveOfUserName );
			}
			// 提交保存
			editor.commit();
		}

		/* 显示 Toast */
		public void DisplayToast(String str) {
			Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
			// 设置 toast 显示的位置
			toast.setGravity(Gravity.TOP, 0, 220);
			// 显示该 toast
			toast.show();
		}
	}
```

4.2.4　Android解析XML的方法

在 Android 开发中，常常需要对 XML 进行操作，特别是解析后台数据的时候，需要将数据封装成 XML 格式类型的文件；而在传递参数给后台程序接口的时候，也需要将数据封装成 XML 格式的文件，这样做便于数据的传递与解析。在 Android 开发平台下，有三种方式解析 XML 文件。

1. DOM方式创建解析XML文档

DOM 方式解析 XML 文件时，会将 XML 文件的所有内容以文档树的方式存放在内存中，然后允许我们使用 DOM API 遍历 XML 树、检索所需的数据。使用 DOM 操作 XML 的代码看起来是比较直观的，并且在某些方面比基于 SAX 方式解析 XML 实现的方式更加简

单。但是，因为 DOM 需要将 XML 文件的所有内容以文档树方式存放在内存中，所以内存的消耗比较大，如果 XML 文件的内容比较小时，采用 DOM 方式是比较合理的。下面介绍 DOM 方式解析 XML 文件的常用属性和方法，如表 4-7 所示。

表 4-7　DOM 方式解析 XML 文件常用属性和方法

DOM 方式解析 XML	功 能 描 述
nodeName	节点对象的名称
nodeValue	节点对象的值
parentNode	节点对象的父节点
childNodes	节点对象的子节点
attributes	节点对象的属性节点
getElementsByTagName(name)	获取带有指定标签名称的所有元素
appendChild(node)	向节点对象中插入子节点
removeChild(node)	从节点对象中删除子节点

下面通过一个示例，介绍如何创建 XML 文档以及如何通过 DOM 方式解析 XML 文档，代码运行结果如图 4-13 和图 4-14 所示。



图 4-13　DOM 方式解析 XML 示例代码运行结果图（1）



图 4-14　DOM 方式解析 XML 示例代码运行结果图（2）

首先，介绍如何通过 XML 布局实现这一效果，如代码清单 4-27 所示。

代码清单 4-27　DOM方式解析XML示例（第 4 章\Demo_04_12）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```



```
        <Button
            android:id="@+id/btn1"
            android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:text="创建 XML 文件"
        />
        <Button
            android:id="@+id/btn2"
            android:layout_width="fill_parent"
           android:layout_height="wrap_content"
           android:text="DOM 解析 XML"
        />
        <TextView
            android:id="@+id/result"
            android:layout_width="fill_parent"
           android:layout_height="wrap_content"
        />
</LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现 DOM 方式解析 XML，如代码清单 4-28 所示。

代码清单 4-28　DOM方式解析XML示例（第 4 章\Demo_04_12）MainActivity.java

```
package com.chen.android;

import java.io.File;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import javax.xml.parsers.DocumentBuilder;
import javax.xml.parsers.DocumentBuilderFactory;
import javax.xml.parsers.ParserConfigurationException;
import org.w3c.dom.Document;
import org.w3c.dom.Element;
import org.w3c.dom.NodeList;
import org.xml.sax.SAXException;
import org.xmlpull.v1.XmlSerializer;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.util.Xml;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;

/**
 * DOM 方式创建与解析 XML
 * @author
```




```
 */
public class MainActivity extends Activity   {
    /*定义 xml 文档存储的位置*/
      private static final String BOOKS_PATH = "/sdcard/books.xml";
      /* 声明 mButton1，mButton2 对象 */
      private Button mButton1,mButton2;
      /* 声明 mTextView 对象 */
      private TextView mTextView;

     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          initViews();
     }
     /**
      * 获取 main.xml 中的控件对象
      */
     private void initViews(){
      mTextView = (TextView)findViewById(R.id.result);
      mButton1 = (Button)findViewById(R.id.btn1);
      mButton2 = (Button)findViewById(R.id.btn2);

      /* 设置 Button 事件监听 */
      mButton1.setOnClickListener(createXmlDocument);
      mButton2.setOnClickListener(parseXmlByDom);
     }

        /**
       * 实现事件监听
       */
     Button.OnClickListener createXmlDocument=new   Button.OnClickListener (){
            @Override
            public void onClick(View v) {
                  // 创建 xml
                  createXmlFile();
            }
     };

     /**
       * 实现事件监听
       */
     Button.OnClickListener parseXmlByDom=new   Button.OnClickListener (){
            @Override
            public void onClick(View v) {
                  // dom 解析 xml
                  domParseXML();
            }
     };

     /**
```



```
  *  创建 xml 文件
  */
private void createXmlFile(){
 /*读取保存的文件*/
 File linceseFile = new File(BOOKS_PATH);
 try{
       linceseFile.createNewFile();
 }catch (IOException e) {
       Log.e("IOException", "exception in createNewFile() method");
       }
 /*初始化文件输出流对象*/
 FileOutputStream fileos = null;
 try{
       fileos = new FileOutputStream(linceseFile);
 }catch (FileNotFoundException    e) {
       Log.e("FileNotFoundException", "can't create FileOutputStream");
       }
 /*将对象序列化到 XML  文档中*/
 XmlSerializer serializer = Xml.newSerializer();
 try {
          serializer.setOutput(fileos,"UTF-8");
          serializer.startDocument(null, true);
          serializer.startTag(null, "books");
          /*循环产生元素*/
          for(int i = 0; i < 3; i ++){
               serializer.startTag(null, "book");
               serializer.startTag(null, "bookname");
               serializer.text("Android 教程" + i);
               serializer.endTag(null, "bookname");
               serializer.startTag(null, "bookauthor");
               serializer.text("Frankie" + i);
               serializer.endTag(null, "bookauthor");
               serializer.endTag(null, "book");
          }
          /*序列化 xml 结束标签*/
          serializer.endTag(null, "books");
          serializer.endDocument();
          serializer.flush();
          fileos.close();
     } catch (Exception e) {
       Log.e("Exception","error occurred while creating xml file");
     }
     Toast.makeText(getApplicationContext(), "创建 xml 文件成功!",
             Toast.LENGTH_SHORT).show();
}

/**
  * dom 解析 xml 文件
  */
private void domParseXML(){
```

```
        File file = new File(BOOKS_PATH);
        /*初始化解析器工厂*/
        DocumentBuilderFactory dbf = DocumentBuilderFactory.newInstance();
        /*初始化文档构造器对象*/
        DocumentBuilder db = null;
        try {
            /*使用当前配置的参数创建一个新的 newInstance 实例，
             * 获取此类的实例之后，将可以从各种输入源解析 XML*/
                db = dbf.newDocumentBuilder();
            } catch (ParserConfigurationException e) {
                e.printStackTrace();
            }
            /*初始化文档对象*/
            Document doc = null;
            try {
                /*将文件转换为 xml 文档*/
                doc = db.parse(file);
            } catch (SAXException e) {
                e.printStackTrace();
            } catch (IOException e) {
                e.printStackTrace();
            }
            /*获取 xml 文档的所有子元素*/
            Element root = doc.getDocumentElement();
            /*获取 xml 文档根元素下标签名为"book"的所有子结点*/
            NodeList books = root.getElementsByTagName("book");
            String res = "本结果是通过 dom 解析:" + "\n";
            /*循环遍历解析节点*/
            for(int i = 0; i < books.getLength();i++){
                Element book = (Element)books.item(i);
                Element bookname =
                    (Element)book.getElementsByTagName("bookname").item(0);
                Element bookauthor =
                    (Element)book.getElementsByTagName("bookauthor").item(0);
                res += "书名: " + bookname.getFirstChild().getNodeValue() + " " +
                        "作者: " + bookauthor.getFirstChild().getNodeValue() + "\n";
            }
            /*将解析的 xml 信息显示在文本控件里*/
            mTextView.setText(res);
        }

    }
```

最后，需要在 AndroidManifest.xml 要添加读写权限，具体如代码清单 4-29 所示。

代码清单 4-29　DOM方式解析XML 示例（第 4 章\Demo_04_12）AndroidManifest.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <manifest xmlns:android="http://schemas.android.com/apk/res/android"
              package="com.chen.android"
```



```
            android:versionCode="1"
            android:versionName="1.0">
        <uses-sdk android:minSdkVersion="8" />

        <application android:icon="@drawable/icon" android:label="@string/app_name">
            <activity android:name=".MainActivity"
                        android:label="@string/app_name">
                <intent-filter>
                    <action android:name="android.intent.action.MAIN" />
                    <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
            </activity>
        </application>
         <uses-permission android:name="android.permission.WRITE_EXTERNAL_STORAGE" />
    </manifest>
```

2. SAX方式解析XML文档

使用 DOM 方式解析 XML 文档，该方式比较符合我们的日常思维方式，容易上手，但是它直接把文档调入内存中，比较耗内存。在这里介绍另一种解析 XML 的方式——SAX（Simple API for XML）方式。SAX 是一个解析速度较快并且占用内存相对少的 XML 解析器，非常适合用于 Android 等移动设备。它主要是基于事件驱动，而 Android 的事件处理机制是基于回调函数。在这里回调函数可以这样理解：在使用 SAX 解析 XML 文档时候，在读取到文档开始和结束标签时就会回调一个事件，在读取到其他节点与内容时也会回调一个事件。它并不需要解析整个文档，在按内容顺序解析文档的过程中，SAX 会判断当前读到的字符是否符合 XML 语法中的某部分，如果符合就会触发事件。

既然涉及事件，就有事件源、事件处理器。所谓事件，其实就是一些回调（callback）方法，这些方法（事件）定义在 ContentHandler 接口中；事件源是在 org.xml.sax 包中的 XMLReader，它通过 parser()方法来解析 XML 文档，并产生事件；事件处理器是 org.xml.sax 包中 ContentHander、DTDHander、ErrorHandler，以及 EntityResolver 这 4 个接口的集合。事件源和事件处理器是如何衔接的呢？主要是通过 XMLReader 相对应的事件处理器注册方法 setXXXX()来完成与 ContentHander、DTDHander、ErrorHandler，以及 EntityResolver 这 4 个接口的连接，四大事件处理器的详细介绍见表 4-8 所示。

表 4-8　SAX 接口中包含的事件处理器及其功能介绍

处理器名称	处 理 事 件	XMLReader 注册方法
ContentHander	跟文档内容有关的事件，可定义： 1）文档的开始与结束 2）XML 元素的开始和结束 3）可忽略的实体 4）名称空间前缀映射的开始和结束 5）处理指令 6）字符数据和可忽略的空格	setContentHander(ContentHander h)
ErrorHandler	处理 XML 文档时产生的错误	setErrorHandler(ErrorHandler　h)
DTDHander	处理对文档的 DTD 进行解析时产生的相应事件	setDTDHander(DTDHander h)
EntityResolver	处理外部实体	setEntityResolver(EntityResolver　h)

在这里我们无需都继承这 4 个接口，SDK 会为我们提供了 DefaultHandler 类来进行处理，DefaultHandler 类的一些主要事件及其功能如表 4-9 所示。

表 4-9　DefaultHandler 类中的主要事件及其功能

方　法　名	功 能 说 明
setDocumentLocator(Locator locator)	设置一个可以定位文档内容事件发生位置的定位对象
startDocument()	用于处理文档解析开始事件
startElement(String namespaceURI，String localName，String qName，Attributes atts)	处理元素开始事件，从参数中可以获取元素所在空间 URL，元素名称，属性列表等信息
characters(char ch[]，int start，int length)	处理元素的字符内容，从参数中可以获得内容
endElement(String namespaceURI，String localName，String qName)	处理元素结束事件，从参数中可以获取元素所在空间 URL，元素名称，属性列表等信息
endDocument()	用于处理文档解析的结束事件

如何通过 SAX 方式解析 XML 呢？在这里我们需要 XmlReader 以及 DefaultHandler 配合起来使用解析 XML。接下来我们总结一下 SAX 解析 XML 文档的处理思路。

1）创建 SAXParserFactory 对象。

2）根据 SAXParserFactory.newSAXParser()方法返回一个 SAXParser 解析器。

3）根据 SAXParser 解析器获取事件源对象 XMLReader。

4）实例化一个 DefaultHandler 对象。

5）连接事件源对象 XMLReader 到事件处理类 DefaultHandler 中。

6）调用 XMLReader 的 parse 方法从输入源中获取到的 XML 数据。

7）通过 DefaultHandler 返回我们需要的数据集合。

下面通过一个示例讲述 SAX 如何解析 XML 的过程，XML 的数据源结构如代码清单 4-30 所示，其中，encoding="utf-8"定义了 XML 的编码格式，有时还可以加 standalone 属性用于说明文档有没有依赖于外面的任何文件，如果有为“YES”，否则为“NO”，此属性为可选，默认为" YES "。该示例的主要功能是：通过 SAX 方式解析 1.xml 文档中的 mytag 节点的值以及解析 tagwithnumber 节点的 thenumber 属性的值，示例代码运行结果如图 4-15 所示。

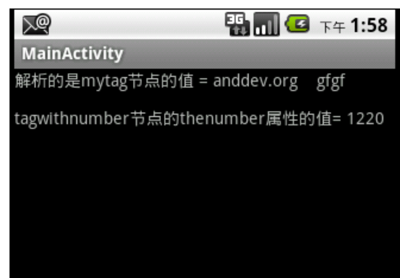


图 4-15　SAX 解析 XML 运行结果

代码清单 4-30　第 4 章SAX解析XML示例（第 4 章\Demo_04_13）数据源 1.xml

```
<?xml version="1.0" encoding="utf-8"?>
<outertag>
  <innertag sampleattribute="innertagAttribute">
        <mytag>anddev.org      gfgf</mytag>
        <tagwithnumber thenumber="1220" />
  </innertag>
</outertag>
```

首先，介绍如何通过 xml 布局实现 SAX 解析 XML 这一效果，如代码清单 4-31 所示。



代码清单 4-31　SAX方式解析XML示例（第 4 章\Demo_04_13）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
   <TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 SAX 方式解析 XML 这一效果，如代码清单 4-32 至代码清单 4-34 所示。

代码清单 4-32　SAX方式解析XML示例（第 4 章\Demo_04_13）MainActivity.java

```
package com.chen.android;

import java.net.URL;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
/**
 * Demo_04_13　SAX 解析 xml 文档
 * @author
 */
public class MainActivity extends Activity {

    private final String MY_DEBUG_TAG = "MyApp";

    /**  当 activity 首次创建时响应调用*/
    @Override
    public void onCreate(Bundle icicle) {
        super.onCreate(icicle);

        /*  创建一个 TextView*/
        TextView tv = new TextView(this);
        try {
            /*  创建一个 url（自己发布的文件地址）
             *  也可以将 1.xml 文件放在项目的 assets 中，从本地路径中读取  */
```

```
                URL url = new URL("http://220.165.15.216:7777/MobilePage/1.xml");

                /* 1、创建 SAXParserFactory 对象*/
                SAXParserFactory spf = SAXParserFactory.newInstance();

                /*2、根据 SAXParserFactory.newSAXParser()方法返回一个 SAXParser 解析器*/
                SAXParser sp = spf.newSAXParser();

                /*3、根据 SAXParser 解析器获取事件源对象 XMLReader*/
                XMLReader xr = sp.getXMLReader();

                /*4、实例化一个 ExampleHandler 对象，该对象继承 DefaultHandler*/
                ExampleHandler myExampleHandler = new ExampleHandler();

                /*5、连接事件源对象 XMLReader 到事件处理类 DefaultHandler 中*/
                xr.setContentHandler(myExampleHandler);

                /* 6、调用 XMLReader 的 parse 方法从输入源中获取到的 xml 数据 */
                xr.parse(new InputSource(url.openStream()));
                /*传递参数完毕. */

                /* 7、通过 DefaultHandler 返回我们需要的数据集合
                 * 将 myExampleHandler 对象传递到自定义的 ParsedExampleDataSet 中*/
                ParsedExampleDataSet parsedExampleDataSet =
                                            myExampleHandler.getParsedData();

                /* 读取数据并赋值 */
                tv.setText(parsedExampleDataSet.toString());

            } catch (Exception e) {
                /* 显示错误信息 */
                tv.setText("Error: " + e.getMessage());
                Log.e(MY_DEBUG_TAG, "WeatherQueryError", e);
            }
            /* 将解析的数据显示到控件上 */
            this.setContentView(tv);
        }
    }
```

代码清单 4-33　SAX方式解析XML示例（第 4 章\Demo_04_13）ExampleHandler.java

```
package com.chen.android;

import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;

/**
 * 自定义的 xml 数据处理类，继承 DefaultHandler
 * @author
```



```
 */
public class ExampleHandler extends DefaultHandler{


        private boolean in_outertag = false;
        private boolean in_innertag = false;
        private boolean in_mytag = false;

        private ParsedExampleDataSet myParsedExampleDataSet
           = new ParsedExampleDataSet();

        /**
         * 获取 ParsedExampleDataSet 对象
         */
        public ParsedExampleDataSet getParsedData() {
                return this.myParsedExampleDataSet;
        }

        /**
         * 回调函数用于文档开始
         * 功能：用于处理文档解析开始时间
         */
        @Override
        public void startDocument() throws SAXException {
                this.myParsedExampleDataSet = new ParsedExampleDataSet();
        }
        /**
         * 回调函数用于文档结束
         * 功能：用于处理文档解析的结束事件
         */
        @Override
        public void endDocument() throws SAXException {
                // 不进行任何处理
        }

         /*
        *   处理元素开始事件
        *      namespaceURI：元素的命名空间
        *      localName ：元素的本地名称（不带前缀）
        *      qName ：元素的限定名（带前缀）
        *      atts ：元素的属性集合
        *      功能： 从参数中可以获取元素所在空间 URL，元素名称，属性列表等信息
        */
        @Override
        public void startElement(String namespaceURI, String localName,
                        String qName, Attributes atts) throws SAXException {
                /*如果节点检索到，则返回 true*/
                if (localName.equals("outertag")) {
                        this.in_outertag = true;
                }else if (localName.equals("innertag")) {
```




```
                this.in_innertag = true;
            }else if (localName.equals("mytag")) {
                this.in_mytag = true;
            }else if (localName.equals("tagwithnumber")) {
                // 提取属性值
                String attrValue = atts.getValue("thenumber");
                int i = Integer.parseInt(attrValue);
                myParsedExampleDataSet.setExtractedInt(i);
            }

        }

        /*
         * 处理元素结束事件
         *     uri ：元素的命名空间
         *     localName ：元素的本地名称（不带前缀）
         *     name ：元素的限定名（带前缀）
         */
        @Override
        public void endElement(String namespaceURI, String localName, String qName)
                throws SAXException {
            if (localName.equals("outertag")) {
                this.in_outertag = false;
            }else if (localName.equals("innertag")) {
                this.in_innertag = false;
            }else if (localName.equals("mytag")) {
                this.in_mytag = false;
            }else if (localName.equals("tagwithnumber")) {
                // 不进行任何处理
            }
        }

        /**
          * 接收字符数据的事件
            */
        @Override
       public void characters(char ch[], int start, int length) {
            //如果读取到的是 mytag 节点
            if(this.in_mytag){
            myParsedExampleDataSet.setExtractedString(new String(ch, start, length));
        }

       }
}
```

代码清单 4-34　SAX方式解析XML示例（第 4 章\Demo_04_13）ParsedExampleDataSet.java

```
package com.chen.android;
```



```
/**
 * 自定义的数据集
 * @author
 */
public class ParsedExampleDataSet {
        private String extractedString = null;
        private int extractedInt = 0;

        public String getExtractedString() {
                return extractedString;
        }
        public void setExtractedString(String extractedString) {
                this.extractedString = extractedString;
        }

        public int getExtractedInt() {
                return extractedInt;
        }
        public void setExtractedInt(int extractedInt) {
                this.extractedInt = extractedInt;
        }

        public String toString(){
                return   "解析的是 mytag 节点的值  = " + this.extractedString+"\n\n"
                        + "tagwithnumber 节点的 thenumber 属性的值= " + this.extractedInt;
        }
}
```

最后，需要在 AndroidManifest.xml 配置文件要添加访问权限，具体如代码清单 4-35 所示。

代码清单 4-35　SAX方式解析XML示例（第 4 章\Demo_04_13）AndroidManifest. xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="org.anddev.android.parsingxml"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name="com.chen.android.MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
   <uses-permission android:name="android.permission.INTERNET" />
    <uses-sdk android:minSdkVersion="8" />
</manifest>
```

3. Pull方式解析XML文档

除了可以使用 SAX 和 DOM 解析 XML 文件，还可以使用 Android 内置的 Pull 解析器解析 XML 文件。 Pull 解析器的运行方式与 SAX 解析器相似，也是提供了类似于事件的处理机制，只不过 PULL 方式读 XML 回调方法返回的是数字。具体解析过程是：在调用开始元素和结束元素事件时，使用 parser.next()可以进入下一个元素并触发相应事件，事件将作为数值代码被发送，可以使用一个 switch 多分支结构对需要的事件进行处理。下面讲述 XmlPullParser 解析器常见属性及方法，如表 4-10 所示。

表 4-10　XmlPullParser 解析器常见属性及方法

方　法　名	功 能 说 明
setInput (InputStream inputStream，　String inputEncoding)	设置输入流及其编码格式
getEventType()	获取当前事件类型
nextText()	如果当前事件是开始标签（START_TAG）则返回的是下一个节点的内容，如果当前事件是结束标签（END_TAG）则返回空值
getName()	获得节点的名字
next()	获取下一个传递过来的事件类型
属性名	
START_DOCUMENT：枚举类型，与 0 等价	读取到 xml 的声明返回
END_DOCUMENT：枚举类型，与 1 等价	读取到 xml 的结束返回
START_TAG：枚举类型，与 2 等价	读取到 xml 的开始标签返回
END_TAG：枚举类型，与 3 等价	读取到 xml 的结束标签返回
TEXT：枚举类型，与 4 等价	读取到 xml 的文本返回

下面通过一个示例具体讲述 Pull 如何解析 XML 的过程，XML 的数据源结构如代码清单 4-36 所示（books.xml）。示例的主要功能是将 books.xml 的节点信息依次解析出来，示例代码运行结果如图 4-16 所示。

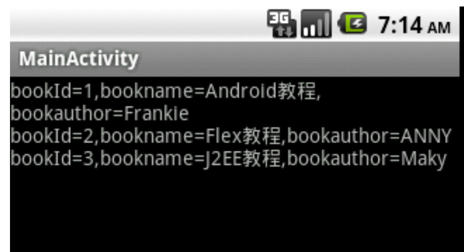


图 4-16　Pull 解析 XML 示例代码运行结果

代码清单 3-36　Pull解析XML示例（第 4 章\Demo_04_14）数据源books.xml

```
<?xml version='1.0' encoding='UTF-8' standalone='yes' ?>
<books>
    <book>
        <bookId>1</bookId>
        <bookname>Android 教程</bookname>
        <bookauthor>Frankie</bookauthor>
    </book>
    <book>
        <bookId>2</bookId>
        <bookname>Flex 教程</bookname>
        <bookauthor>ANNY</bookauthor>
    </book>
```



```
        <book>
                <bookId>3</bookId>
                <bookname>J2EE 教程</bookname>
                <bookauthor>Maky</bookauthor>
        </book>
</books>
```

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-37 所示。

代码清单 4-37　Pull解析XML示例（第 4 章\Demo_04_14）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 Pull 解析 XML，如代码清单 4-38 至代码清单 4-40 所示。

代码清单 4-38　Pull解析XML示例（第 4 章\Demo_04_14）MainActivity.java

```
package com.chen.android;

import java.util.List;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
/**
 * Demo_04_14 Pull 解析 XML 文档
 * @author
 */
public class MainActivity extends Activity {
        /*定义一个常量*/
        private final String MY_DEBUG_TAG = "MainActivity";
        /** 定义一个变量 */
        private String bookInfo="";

        @Override
        public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
```

```
        /* 创建一个文本接收解析到的数据 */
        TextView tv = new TextView(this);

        try {
            /* 从 url 地址里读取数据（可从 assets 中读取，
             * 也可以自己把 books.xml 发布到服务器里）*/
            String url="http://220.165.15.216:7777/MobilePage/books.xml";
            List<Books> books = XmlPull.URLReadXmlByPull(url);
            for(Books book : books){
                bookInfo+=book.toString();
            }

            /*将解析到的书信息显示在 TextView 控件里*/
            tv.setText(bookInfo);

        } catch (Exception e) {
            /* 处理异常 */
            tv.setText("Error: " + e.getMessage());
            Log.e(MY_DEBUG_TAG, "WeatherQueryError", e);
        }
        /* 将定义的控件显示在屏幕上 */
        this.setContentView(tv);
    }
}
```

代码清单 4-39　第 4 章Pull解析XML 示例（第 4 章\Demo_04_14）XmlPull.java

```
package com.chen.android;

import java.net.URL;
import java.util.ArrayList;
import java.util.List;
import org.xmlpull.v1.XmlPullParser;
import org.xmlpull.v1.XmlPullParserFactory;
import android.util.Log;
/**
 * pull 方式解析 xml 类
 * @author
 */
public class XmlPull  {
    /**
     * 从网络地址 URL 读取 XML 文件用 Pull 解析 XML
     * @param xmlUrlPath xml 的网络地址
     * @return BooksList
     * @throws Exception
     */
    public static List<Books> URLReadXmlByPull(String xmlUrlPath)
                                                throws Exception{
        /*定义一个 List，实现 ArrayList，从来装载 n 本书的信息*/
        List<Books> listBooks =  new ArrayList<Books>();
```



```
/*初始化定义的书类*/
Books books = null;
/*将用户定义的 url 装载在 URL 类中*/
URL url = new URL(xmlUrlPath);
//构建 XmlPullParserFactory
XmlPullParserFactory pullParserFactory
            = XmlPullParserFactory.newInstance();
//获取 XmlPullParser 的实例
XmlPullParser xmlPullParser = pullParserFactory.newPullParser();
/*写日志*/
Log.i("PullParseXML", "getXML......");
/*设置输入流，读入 xml 文件信息流*/
xmlPullParser.setInput(url.openConnection().getInputStream(), "UTF-8");
/*写日志，开始解析*/
Log.i("PullParseXML", "PullParseXML....start....");
/**获取事件类型
 * pull 读到 xml 后返回数字
 *    读取到 xml 的声明返回数字 0 START_DOCUMENT
       读取到 xml 的结束返回数字 1 END_DOCUMENT
       读取到 xml 的开始标签返回数字 2 START_TAG
       读取到 xml 的结束标签返回数字 3 END_TAG
       读取到 xml 的文本返回数字 4 TEXT
 */
int eventType=xmlPullParser.getEventType();
/**
 * 只要这个事件返回的不是 1（文件未结束），就一直读取 xml 文件
 */
while(eventType != XmlPullParser.END_DOCUMENT){
    /*获得节点的名字*/
    String nodeName=xmlPullParser.getName();
    /*判断 xmlPullParser 事件类型*/
    switch (eventType) {
    /*如果是文档开始，则退出*/
    case XmlPullParser.START_DOCUMENT:
        break;
        /*如果事件是标签开始，则判断节点名称是否有匹配对象*/
    case XmlPullParser.START_TAG:
        /*如果存在"book"节点*/
        if("book".equals(nodeName)){
            /*则初始化 books 对象*/
            books = new Books();
        }
        /*如果存在"bookId"节点*/
        if("bookId".equals(nodeName) && books != null){
            /*则往 books 对象中添加 bookid 元素*/
            books.setBookId(xmlPullParser.nextText());
        }
        /*如果存在"bookname"节点*/
        if("bookname".equals(nodeName)   && books != null){
            /*则往 books 对象中添加 bookname 元素*/
```

```
                books.setBookname(xmlPullParser.nextText());
            }
            /*如果存在"bookauthor"节点*/
            if("bookauthor".equals(nodeName) && books != null){
                /*则往 books 对象中添加 bookauthor 元素*/
                books.setBookauthor(xmlPullParser.nextText());
            }
            break;
            /*如果事件是标签结束*/
        case XmlPullParser.END_TAG:
            /*找到结束标签*/
            if("book".equals(nodeName)){
                /*然后往 list 中添加 books*/
                listBooks.add(books);
            }
            break;
        default:
            break;
        }
        /*读取下一个事件类型*/
        eventType = xmlPullParser.next();
    }
    /*访回 listBooks*/
    return listBooks;
  }
}
```

代码清单 4-40　第 4 章Pull解析XML 示例（第 4 章\Demo_04_14）Books.java

```
package com.chen.android;
/**
 *  书子类
 * @author
 */
public class Books {
    private String bookId;    //书 ID
    private String bookname; //书名
    private String bookauthor; //书作者

    /*get、set 封装 bookId、bookname、bookname 属性*/
    public String getBookId() {
        return bookId;
    }
    public void setBookId(String bookId) {
        this.bookId = bookId;
    }
    public String getBookname() {
        return bookname;
    }
    public void setBookname(String bookname) {
```



```
            this.bookname = bookname;
        }
        public String getBookauthor() {
            return bookauthor;
        }
        public void setBookauthor(String bookauthor) {
            this.bookauthor = bookauthor;
        }
        /**
         * 重写 toString()方法
         * 将结果拼接成字符串返回
         */
        @Override
        public String toString() {
            return "bookId=" + bookId + ",bookname=" + bookname +
            ",bookauthor=" + bookauthor+"\n";
        }
    }
```

最后，需要在 AndroidManifest.xml 要添加访问权限，具体如代码清单 4-41 所示。

代码清单 4-41　第 4 章Pull解析XML 示例（第 4 章\Demo_04_14）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name="com.chen.android.MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-sdk android:minSdkVersion="8" />
</manifest>
```

4.2.5　Android如何与后台通信

与后台通信在本书中是指 Android 如何与后台 Web 数据进行交互，具体包括与后台进行增、删、改、查等操作。虽然 Android 提供了 SQLite（在后面的章节中涉及）可以直接在 Android 项目中直接建小型数据库，对数据进行增、删、改、查的操作，但是由于手机内存容量及性能的限制、SQLite 的功能不完善，在 Android 应用程序中进行大量处理数据是不太合理的，因此提倡通过网络连接与后台交互的方式，通过在后台创建处理程序，然后与前端进行通信。

Android 与后台通信的方式有多种，可以通过 Http 通信，还可以通过 Socket 通信。Http 通信是基于 HTTP 协议，该协议可以说是当今 Internet 上使用最多、最重要的协议，越来越多的 Java 应用程序需要直接通过 HTTP 协议来访问网络资源。有关 HTTP 通信的包存在于 JDK 的 java.net 包中，在该包中提供了访问 HTTP 协议的基本功能。而 Socket 通信方式属于套接字通信方式。下面介绍几种基于 Http 和 Socket 的 Android 应用程序通信方式，具体有：不带参数（直接获取数据）HttpURLConnection 与后台通信、带参数 HttpURLConnection GET 方式、带参数 HttpURLConnection POST 方式、带参数 HttpClient 方式（GET 和 POST 方式）及 Socket 方式。

1. 不带参数HttpURLConnection通信方式

不带参数 HttpURLConnection 如何与后台程序进行交互呢，这主要是如何从后台程序中直接获取数据的过程，在获取数据的过程中不带任何处理参数。这是最简单的通信方式，例如查询所有信息或者获取后台的某一个 XML 文件，就可以使用不带任何参数的 HttpURLConnection 通信方式。下面介绍 HttpURLConnection 通信的主要步骤。

1）构造一个 URL 对象，如：

```
URL     url = new URL(httpUrl);
```

2）使用 HttpURLConnection 打开连接，如：

```
HttpURLConnection httpURLConnection = (HttpURLConnection) url.openConnection();
```

3）读取数据流，如：

```
InputStreamReader inputStreamReader = new InputStreamReader(httpURLConnection. getInputStream());
```

4）为输出流创建 BufferedReader，如：

```
BufferedReader bufferedReader = new BufferedReader(inputStreamReader);
```

5）使用循环读取流传过来的数据，如：

```
while (((inputLine = bufferedReader.readLine()) != null))
                {
                    //在每一行后面加上一个"\n"来换行
                    resultData += inputLine + "\n";
                }
```

6）关闭 InputStreamReader 和 HttpURLConnection，如：

```
inputStreamReader.close();
httpURLConnection.disconnect();
```

下面通过一个示例，讲述不带参数 HttpURLConnection 通信的具体用法，示例代码运行结果如图 4-17 所示。



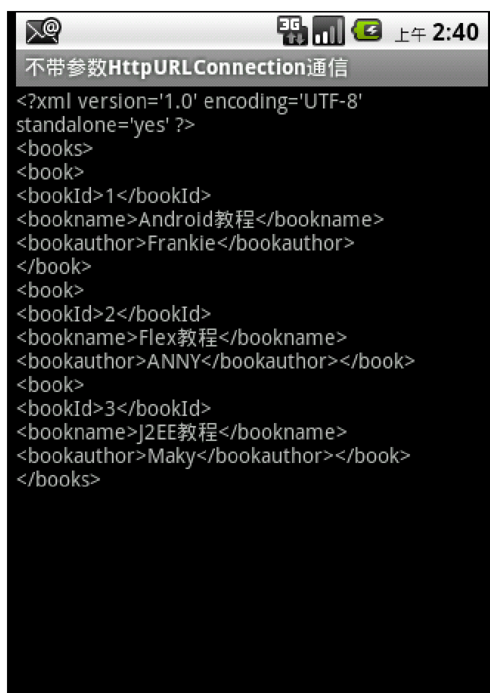


图 4-17　不带参数 HttpURLConnection 与后台通信示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-42 所示。

代码清单 4-42　不带参数HttpURLConnection通信示例（第 4 章\Demo_04_15）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
        android:id="@+id/TextView_HTTP"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 HttpURLConnection 通信，如代码清单 4-43 所示。

代码清单 4-43　不带参数HttpURLConnection通信示例（第 4 章\Demo_04_15）MainActivity.java

```
package com.chen.android;
```




```
import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;

/**
 * 不带参数 HttpURLConnection 与后台通信
 *
 * @author
 */
public class MainActivity extends Activity {
        private final String DEBUG_TAG = "MainActivity";

        /**当 activity 首次创建时响应调用 */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                TextView mTextView = (TextView) this.findViewById(R.id.TextView_HTTP);
                // http 地址
                String httpUrl = "http://220.165.15.216:7777/MobilePage/books.xml";
                // 获得的数据
                String resultData = "";
                URL url = null;
                try {
                        // 构造一个 URL 对象
                        url = new URL(httpUrl);
                } catch (MalformedURLException e) {
                        Log.e(DEBUG_TAG, "MalformedURLException");
                }
                if (url != null) {
                        try {
                                // 使用 HttpURLConnection 打开连接
                                HttpURLConnection httpURLConnection = (HttpURLConnection) url
                                                .openConnection();
                                // 得到读取的内容(流)
                                InputStreamReader inputStreamReader = new InputStreamReader(
                                                httpURLConnection.getInputStream());
                                // 为输出创建 BufferedReader
                                BufferedReader bufferedReader = new BufferedReader(
                                                inputStreamReader);
                                String inputLine = null;
```



```
                // 使用循环来读取获得的数据
                while (((inputLine = bufferedReader.readLine()) != null)) {
                    // 我们在每一行后面加上一个"\n"来换行
                    resultData += inputLine + "\n";
                }
                // 关闭 InputStreamReader
                inputStreamReader.close();
                // 关闭 http 连接
                httpURLConnection.disconnect();
                // 设置显示取得的内容
                if (resultData != null) {
                    mTextView.setText(resultData);
                } else {
                    mTextView.setText("读取的内容为 NULL");
                }
            } catch (IOException e) {
                Log.e(DEBUG_TAG, "IOException");
            }
        } else {
            Log.e(DEBUG_TAG, "Url NULL");
        }
    }
}
```

最后，需要在 AndroidManifest.xml 要添加访问权限，如代码清单 4-44 所示。

代码清单 4-44　不带参数HttpURLConnection通信示例（第 4 章\Demo_04_15）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
     <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

2. HttpURLConnection GET通信方式

有一定编程基础的读者都知道，带参数的 HttpURLConnection 通信方式有两种，这两种

方式分别是 GET 和 POST，GET 请求可以获取静态页面，也可以把参数放在 URL 字符串后面，它处理的数据量相对较少。使用这种通信方式，如果参数中有中文，则容易出现乱码，导致参数无法正确地被后台程序所处理。而 POST 不是放在 URL 字符串后面，而是放在 Http 请求的正文内。相对而言，处理的参数数据量比较大，可以通过设置编码方式及对参数进行封装，能处理中文且不易乱码，且在传输中相对安全。因此，POST 和 GET 的主要区别在于请求方式的不同，GET 可以获得静态页面，也可以把参数放在 URL 字符串后面，传递给服务器，而 POST 方法的参数则是放在 Http 请求内容中。

下面通过一个示例，讲述 HttpURLConnection GET 通信方式，示例代码运行结果如图 4-18 所示。



图 4-18　HttpURLConnection GET 方式与后台通信运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-45 所示。

代码清单 4-45　HttpURLConnection GET通信方式示例（第 4 章\Demo_04_16）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
        android:id="@+id/TextView_HTTP"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
    />
</LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现 HttpURLConnection 通信，如代码清单 4-46 所示。

代码清单 4-46　HttpURLConnection GET通信方式示例（第 4 章\Demo_04_16）MainActivity.java

```
package com.chen.android;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
```



```
import java.net.MalformedURLException;
import java.net.URL;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.widget.TextView;
/**
 * 带参数 HttpURLConnection GET 通信方式
 * @author
 */
public class MainActivity extends Activity
{

    private final String DEBUG_TAG = "MainActivity";
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        TextView mTextView = (TextView)this.findViewById(R.id.TextView_HTTP);
        //http 是地址，"?strName=abcdefg"是作者上传的参数
        String httpUrl = "http://220.165.15.216:7777/MobilePage/LoginHtmlPage.aspx"
                        +"?strName=000377&strPwd=123456";
        //初始化获得的数据
        String resultData = "";
        URL url = null;
        try
        {
            //构造一个 URL 对象
            url = new URL(httpUrl);
        }
        catch (MalformedURLException e)
        {
            Log.e(DEBUG_TAG, "MalformedURLException");
        }
        if (url != null)
        {
            try {
                // 使用 HttpURLConnection 打开连接
                HttpURLConnection httpURLConnection = (HttpURLConnection) url
                        .openConnection();
                // 得到读取的内容(流)
                InputStreamReader inputStreamReader = new InputStreamReader(
                        httpURLConnection.getInputStream());
                // 为输出创建 BufferedReader
                BufferedReader bufferedReader = new BufferedReader(
                        inputStreamReader);
                String inputLine = null;
                // 使用循环来读取获得的数据
```

```
                    while (((inputLine = bufferedReader.readLine()) != null)) {
                        // 在每一行后面加上一个"\n"来换行
                        resultData += inputLine + "\n";
                    }
                    // 关闭 InputStreamReader
                    inputStreamReader.close();
                    // 关闭 http 连接
                    httpURLConnection.disconnect();
                    // 设置显示取得的内容
                    if (resultData != null) {
                        mTextView.setText(resultData);
                    } else {
                        mTextView.setText("读取的内容为空");
                    }
                } catch (IOException e) {
                    Log.e(DEBUG_TAG, "IOException");
                }
            }
            else
            {
                Log.e(DEBUG_TAG, "Url NULL");
            }

        }
    }
```

最后，需要在 AndroidManifest.xml 要添加访问权限，具体如代码清单 4-47 所示。

代码清单 4-47　HttpURLConnection GET通信方式示例（第 4 章\Demo_04_16）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
     <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```



3. HttpURLConnection POST通信方式

HttpURLConnection POST 通信方式在上一节已经阐述过，在这里需要注意的是：在通信过程中需要将 HttpURLConnection. setDoOutput()和 HttpURLConnection. setDoInput()设置为 true。这一节重点讲解 HttpURLConnection POST 通信方式的参数名和参数值如何放在正文中被处理，在这里需要使用到 URLEncoder.encode(String s，String encoding)对参数进行封装，第一个参数是参数的变量名，第二个参数是编码格式。把封装好的参数通过 DataOutputStream. writeBytes()写进流中。到本节为止，讲解了 HttpURLConnection 三种通信方式，大家可以把三个小程序合并成一个程序，比较 HttpURLConnection 三种通信方式的处理过程的区别。示例代码运行结果如图 4-19、图 4-20 所示。



图 4-19　HttpURLConnection 通信方式运行结果（1）

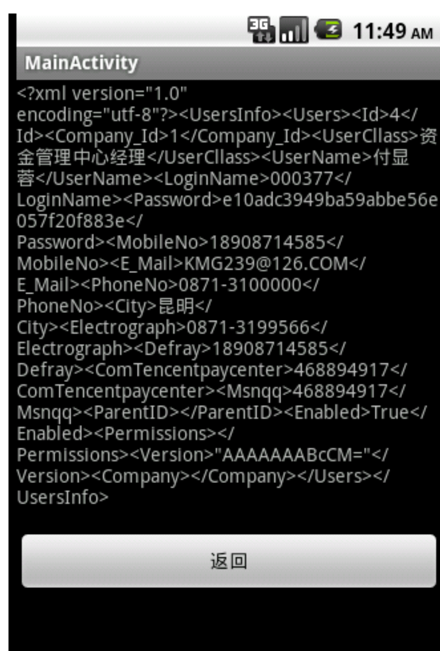


图 4-20　HttpURLConnection 通信方式运行结果（2）

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-48、4-49 所示（可直接使用上一节的代码）。

代码清单 4-48　HttpURLConnection POST通信方式示例（第 4 章\Demo_04_17）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
```

```
            android:text="单击按钮实现不同HttpURLConnection 通信方式"/>
        <Button
            android:id="@+id/Button_HTTP"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="不带参数HttpURLConnection 通信"/>
        <Button
            android:id="@+id/Button_Get"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="带参数GET HttpURLConnection 通信"/>
        <Button
            android:id="@+id/Button_Post"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="带参数POST HttpURLConnection 通信"/>
</LinearLayout>
```

代码清单 4-49　HttpURLConnection POST通信方式示例（第 4 章\Demo_04_17）http.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
      <TextView
     android:id="@+id/TextView_HTTP"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     />
      <Button
            android:id="@+id/Button_Back"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="返回"/>
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 HttpURLConnection 通信，如代码清单 4-50、4-51 所示。

代码清单 4-50　HttpURLConnection POST通信方式示例（第 4 章\Demo_04_17）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
```



```
import android.view.View;
import android.widget.Button;

/**
  * HttpURLConnection  三种通信方式
  * @author
  */
public class MainActivity extends Activity
{
        /**  当 activity 首次创建时响应调用  */
        @Override
        public void onCreate(Bundle savedInstanceState)
        {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                /*获取页面控件元素*/
                Button button_http = (Button) findViewById(R.id.Button_HTTP);
                Button button_Get = (Button) findViewById(R.id.Button_Get);
                Button button_Post = (Button) findViewById(R.id.Button_Post);
                /*  监听 button 的事件信息  */
                button_http.setOnClickListener(GetDateByNoParameter) ;
                button_Get.setOnClickListener(GetDateByGet );
                button_Post.setOnClickListener(GetDateByPost) ;

        }
        /**
          *  实现 button 的事件监听
          */
        Button.OnClickListener    GetDateByNoParameter=new    Button.OnClickListener(){

                @Override
                public void onClick(View arg0) {
                        /*  新建一个 Intent 对象  */
                        Intent intent = new Intent();
                        /*  指定 intent 要启动的类  */
                        intent.setClass(MainActivity.this, Activity02.class);
                        /*  启动一个新的 Activity */
                        startActivity(intent);
                        /*  关闭当前的 Activity */
                        MainActivity.this.finish();
                }
        };

        /**
          *  实现 button 的事件监听
          */
        Button.OnClickListener    GetDateByGet=new    Button.OnClickListener(){

                @Override
```




```
            public void onClick(View arg0) {
                /*  新建一个 Intent 对象  */
                Intent intent = new Intent();
                /*  指定 intent 要启动的类  */
                intent.setClass(MainActivity.this, Activity03.class);
                /*  启动一个新的 Activity */
                startActivity(intent);
                /*  关闭当前的 Activity */
                MainActivity.this.finish();
            }

        };

        /**
         *  实现 button 的事件监听
         */
        Button.OnClickListener   GetDateByPost=new   Button.OnClickListener(){

            @Override
            public void onClick(View arg0) {
                /*  新建一个 Intent 对象  */
                Intent intent = new Intent();
                /*  指定 intent 要启动的类  */
                intent.setClass(MainActivity.this, Activity04.class);
                /*  启动一个新的 Activity */
                startActivity(intent);
                /*  关闭当前的 Activity */
                MainActivity.this.finish();
            }

        };
    }
```

代码清单 4-51　HttpURLConnection POST通信方式示例（第 4 章\Demo_04_17）Activity04.java

```
package com.chen.android;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import java.net.URLEncoder;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
```



```
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

/**
 * 以 POST 方式上传参数
 */
public class Activity04 extends Activity {
	private final String DEBUG_TAG = "Activity04";

	/**当 activity 首次创建时响应调用 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.http);

		TextView mTextView = (TextView) this.findViewById(R.id.TextView_HTTP);
		// http 地址
		String httpUrl = "http://220.165.15.216:7777/MobilePage/LoginHtmlPage.aspx";
		// 获得的数据
		String resultData = "";
		URL url = null;
		try {
			// 构造一个 URL 对象
			url = new URL(httpUrl);
		} catch (MalformedURLException e) {
			Log.e(DEBUG_TAG, "MalformedURLException");
		}
		if (url != null) {
			try {
				// 使用 HttpURLConnection 打开连接
				HttpURLConnection urlConn = (HttpURLConnection) url
						.openConnection();
				// 因为这个是 post 请求，需要设置为 true
				urlConn.setDoOutput(true);
				urlConn.setDoInput(true);
				// 设置以 POST 方式
				urlConn.setRequestMethod("POST");
				// Post 请求不能使用缓存
				urlConn.setUseCaches(false);
				// URLConnection.setInstanceFollowRedirects 是成员函数，仅作用于当前函数
				urlConn.setInstanceFollowRedirects(true);
				// 配置本次连接的 Content-type，配置为 application/x-www-form-urlencoded
				// urlencoded 编码过的 form 参数，下面可以看到对正文内容使用
URLEncoder.encode
				// 进行编码
				urlConn.setRequestProperty("Content-Type",
						"application/x-www-form-urlencoded");
				// 连接，从 postUrl.openConnection()至此的配置必须要在 connect 之前完成，
```




```
                // 要注意的是 connection.getOutputStream 会隐含的进行 connect。
                urlConn.connect();
                // DataOutputStream 流
                DataOutputStream out = new DataOutputStream(
                        urlConn.getOutputStream());
                // 要上传的参数
                String content = URLEncoder.encode("strName", "UTF-8") + "="
                        + URLEncoder.encode("000377", "UTF-8");
                content += "&" + URLEncoder.encode("strPwd", "UTF-8") + "="
                        + URLEncoder.encode("123456", "UTF-8");
                // 将要上传的内容写入流中
                out.writeBytes(content);
                // 刷新、关闭
                out.flush();
                out.close();
                // 获取数据
                BufferedReader reader = new BufferedReader(
                        new InputStreamReader(urlConn.getInputStream()));
                String inputLine = null;
                // 使用循环来读取获得的数据
                while (((inputLine = reader.readLine()) != null)) {
                    // 在每一行后面加上一个"\n"来换行
                    resultData += inputLine + "\n";
                }
                reader.close();
                // 关闭 http 连接
                urlConn.disconnect();
                // 设置显示取得的内容
                if (resultData != null) {
                    mTextView.setText(resultData);
                } else {
                    mTextView.setText("读取的内容为 NULL");
                }
            } catch (IOException e) {
                Log.e(DEBUG_TAG, "IOException");
            }
        } else {
            Log.e(DEBUG_TAG, "Url NULL");
        }

        Button button_Back = (Button) findViewById(R.id.Button_Back);
        /* 监听 button 的事件信息 */
        button_Back.setOnClickListener(new Button.OnClickListener() {
            public void onClick(View v) {
                /* 新建一个 Intent 对象 */
                Intent intent = new Intent();
                /* 指定 intent 要启动的类 */
                intent.setClass(Activity04.this, MainActivity.class);
                /* 启动一个新的 Activity */
                startActivity(intent);
```



```
                        /* 关闭当前的 Activity */
                        Activity04.this.finish();
                    }
                });
            }
        }
```

最后，需要在 AndroidManifest.xml 要添加访问权限，具体如代码清单 4-52 所示。

代码清单 4-52　HttpURLConnection POST通信方式示例（第 4 章\Demo_04_17）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <activity android:name="Activity02"></activity>
        <activity android:name="Activity03"></activity>
        <activity android:name="Activity04"></activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

4. HttpClient 通信方式（GET、POST）

虽然在 JDK 的 java.net 包中已经提供了访问 HTTP 协议的基本功能，但是对于大部分应用程序来说，JDK 库本身所提供的功能还不够丰富和灵活。HttpClient 是 Apache Jakarta Common 下的子项目，用来提供高效的、最新的、功能丰富的支持 HTTP 协议的客户端编程工具包，并且支持最新版本的 HTTP 协议。下面介绍 HttpClient 通信的主要步骤。

1）创建 HttpGet 连接对象，如：

```
HttpGet httpRequest = new HttpGet(url);
```

2）创建 HttpClient 实例，如：

```
HttpClient httpclient = new DefaultHttpClient();
```

3）请求 HttpClient，取得 HttpResponse，如：

```
HttpResponse httpResponse = httpclient.execute(httpRequest);
```

4）判断 Http 响应请求是否成功，如：

```
httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK ;
```

5）读 response，返回数据集结果，如：

```
httpResponse.getEntity() ;
```

6）释放连接，无论执行方法是否成功，都必须释放连接。

下面通过一个示例，讲述 HttpClient GET 和 POST 的用法，示例代码运行结果如图 4-21 和图 4-22 所示。



图 4-21　HttpClient 方式与后台通信运行结果



图 4-22　HttpClient 通信（GET、POST）示例运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 4-53、4-54 所示。

代码清单 4-53　HttpClient通信（GET、POST）示例（第 4 章\Demo_04_18）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="使用 HttpClient 来进行 GET 和 POST 连接"
    />
```



```
    <Button
        android:id="@+id/Button_Get"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="以 HttpClient GET 方式传递数据"/>
    <Button
        android:id="@+id/Button_Post"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="以 HttpClient POST 方式传递数据"/>
</LinearLayout>
```

代码清单 4-54　HttpClient通信（GET、POST）示例（第 4 章\Demo_04_18）http.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
     <TextView
    android:id="@+id/TextView_HTTP"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    />
     <Button
        android:id="@+id/Button_Back"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="返回"/>
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 HttpClient 通信方式，如代码清单 4-55 至代码清单 4-57 所示。

代码清单 4-55　HttpClient通信（GET、POST）示例（第 4 章\Demo_04_18）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

/**
 * HttpClient 方式（Get 和 Post）
 * @author
 */
```




```
public class MainActivity extends Activity
{
    /** 当 activity 首次创建时响应调用 */
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
      /*获取页面的 Button 对象*/
        Button button_Get = (Button) findViewById(R.id.Button_Get);
        Button button_Post = (Button) findViewById(R.id.Button_Post);
        /* 监听 button 的事件信息 */
        button_Get.setOnClickListener(GetDateByGet) ;
        button_Post.setOnClickListener(GetDateByPost );

    }
    /**
     * 实现 button_Get 的事件监听
     */
    Button.OnClickListener    GetDateByGet=new    Button.OnClickListener(){

        @Override
        public void onClick(View arg0) {
            /* 新建一个 Intent 对象 */
            Intent intent = new Intent();
            /* 指定 intent 要启动的类 */
            intent.setClass(MainActivity.this, Activity02.class);
            /* 启动一个新的 Activity */
            startActivity(intent);
            /* 关闭当前的 Activity */
            MainActivity.this.finish();
        }
    };

    /**
     * 实现 button_Post 的事件监听
     */
    Button.OnClickListener    GetDateByPost=new    Button.OnClickListener(){

        @Override
        public void onClick(View arg0) {
            /* 新建一个 Intent 对象 */
            Intent intent = new Intent();
            /* 指定 intent 要启动的类 */
            intent.setClass(MainActivity.this, Activity03.class);
            /* 启动一个新的 Activity */
            startActivity(intent);
            /* 关闭当前的 Activity */
            MainActivity.this.finish();
        }
    };
```



```
    }
```

代码清单 4-56　HttpClient通信（GET、POST）示例（第 4 章\Demo_04_18）Activity02.java

```
package com.chen.android;

import java.io.IOException;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.methods.HttpGet;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.util.EntityUtils;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

/**
 * HttpClient Get 方式
 */
public class Activity02 extends Activity {
    /* 初始化控件 */
    private TextView mTextView;
    private Button button_Back;
    private String erroStr;
    private String httpUrl ;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.http);
        // 获得 xml 布局中的文本控件
        mTextView = (TextView) this.findViewById(R.id.TextView_HTTP);
        // 设置按键事件监听
        button_Back = (Button) findViewById(R.id.Button_Back);
        // HTTP 通信地址+参数
         httpUrl = "http://220.165.15.216:7777/MobilePage/LoginHtmlPage.aspx?"
                 + "strName=000377&strPwd=123456";
        // HttpGet 连接对象
        HttpGet httpRequest = new HttpGet(httpUrl);
        try {
            // 取得 HttpClient 对象
            HttpClient httpClient = new DefaultHttpClient();
```




```
                    // 请求 HttpClient，取得 HttpResponse
                    HttpResponse httpResponse = httpClient.execute(httpRequest);

                    // Http 响应请求成功
                    if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {

                        // 取得返回的字符串
                        String strResult = EntityUtils.toString(httpResponse
                                .getEntity());

                        // 给控件赋值
                        mTextView.setText(strResult);
                    } else {
                        erroStr = "HttpClient 请求失败!";
                        mTextView.setText(erroStr);
                    }
                }
                // 捕获异常
                catch (ClientProtocolException e) {
                    erroStr = e.getMessage().toString();
                    mTextView.setText(erroStr);

                } catch (IOException e) {

                    erroStr = e.getMessage().toString();
                    mTextView.setText(erroStr);
                } catch (Exception e) {

                    erroStr = e.getMessage().toString();
                    mTextView.setText(erroStr);
                }

                /* 监听 button 的事件信息 */
                button_Back.setOnClickListener(returnMainActivity);
            }

            /**
             * 监听方法实现
             */
            Button.OnClickListener returnMainActivity = new Button.OnClickListener() {
                @Override
                public void onClick(View v) {

                    /* 新建一个 Intent 对象 */
                    Intent intent = new Intent();
                    /* 指定 intent 要启动的类 */
                    intent.setClass(Activity02.this, MainActivity.class);
                    /* 启动一个新的 Activity */
                    startActivity(intent);
                    /* 关闭当前的 Activity */
```



```
                    Activity02.this.finish();

                }

            };
}
```

代码清单 4-57　HttpClient通信（GET、POST）示例（第 4 章\Demo_04_18）Activity03.java

```
package com.chen.android;

import java.io.IOException;
import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import java.util.Map;
import org.apache.http.HttpEntity;
import org.apache.http.HttpResponse;
import org.apache.http.HttpStatus;
import org.apache.http.client.ClientProtocolException;
import org.apache.http.client.HttpClient;
import org.apache.http.client.entity.UrlEncodedFormEntity;
import org.apache.http.client.methods.HttpPost;
import org.apache.http.impl.client.DefaultHttpClient;
import org.apache.http.message.BasicNameValuePair;
import org.apache.http.util.EntityUtils;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;

/**
 * HttpClient Post 方式
 */
public class Activity03 extends Activity {
    private String erroStr;
    private TextView mTextView ;
    private     Button button_Back ;
    private String httpUrl ;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.http);
        mTextView= (TextView) this.findViewById(R.id.TextView_HTTP);
        //HTTP 通信地址
```




```
httpUrl= "http://220.165.15.216:7777/MobilePage/LoginHtmlPage.aspx";

// HttpPost 连接对象
HttpPost httpRequest = new HttpPost(httpUrl);
//通过 Map 键值对应处理参数
Map<String, String> map = new HashMap<String, String>();
map.put("strName", "000377");
map.put("strPwd", "123456");

// 使用 NameValuePair 来保存要传递的 Post 参数
List<BasicNameValuePair> postData = new ArrayList<BasicNameValuePair>();
//循环遍历 Map 中的参数,
for (Map.Entry<String, String> entry : map.entrySet()) {
        //将参数添加到 BasicNameValuePair 类型的 List 中
        postData.add(new BasicNameValuePair(entry.getKey(), entry
                    .getValue()));
}
try {
        // 设置字符集 utf-8，还可以设置成 gb2312（简体中文）
        HttpEntity httpentity = new UrlEncodedFormEntity(postData, "utf-8");
        // 请求 httpRequest
        httpRequest.setEntity(httpentity);
        // 取得默认的 HttpClient
        HttpClient httpclient = new DefaultHttpClient();
        // 取得 HttpResponse
        HttpResponse httpResponse = httpclient.execute(httpRequest);
        // HttpStatus.SC_OK 表示连接成功
        if (httpResponse.getStatusLine().getStatusCode() == HttpStatus.SC_OK) {
            // 取得返回的字符串
            String strResult = EntityUtils.toString(httpResponse
                        .getEntity());
            mTextView.setText(strResult);
        } else {
            erroStr="HttpClient Post 方式,请求错误!";
            mTextView.setText(erroStr);
        }
} catch (ClientProtocolException e) {
        erroStr=e.getMessage().toString();
        mTextView.setText(erroStr);

} catch (IOException e) {
        erroStr=e.getMessage().toString();
        mTextView.setText(erroStr);

} catch (Exception e) {
        erroStr=e.getMessage().toString();
        mTextView.setText(erroStr);

}
// 设置按键事件监听
  button_Back = (Button) findViewById(R.id.Button_Back);
/* 监听 button 的事件信息 */
```



```
                    button_Back.setOnClickListener(new Button.OnClickListener() {
                         @Override
                         public void onClick(View v) {
                              /*  新建一个 Intent 对象  */
                              Intent intent = new Intent();
                              /*  指定 intent 要启动的类  */
                              intent.setClass(Activity03.this, MainActivity.class);
                              /*  启动一个新的 Activity */
                              startActivity(intent);
                              /*  关闭当前的 Activity */
                              Activity03.this.finish();
                         }
                    });
               }
          }
```

最后，需要在 AndroidManifest.xml 要添加访问权限，具体如代码清单 4−58 所示。

代码清单 4−58　HttpClient通信（GET、POST）示例（第 4 章\Demo_04_18）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
       <activity android:name="Activity02"></activity>
          <activity android:name="Activity03"></activity>
    </application>
     <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

5. Socket方式

应用程序可以通过 Socket（套接字）来发送和接收数据，这就好像应用程序打开了一个文件句柄，可以将数据读写到稳定的存储器上一样。使用 Socket 可以将应用程序添加到网络中，并与处于同一网络中的其他应用程序进行通信。Socket 包括流套接字（Stream Socket）和数据报套接字（Datagram Socket）两种通信方式。流套接字将 TCP 作为其端对端协议，提供了一个可信赖的字节流服务。而数据报套接字使用 UDP 协议，提供了一个“尽力而为”的数据报服务，应用程序可以通过它发送最长 65500 字节的个人信息。Socket 通信的原理图如图 4−23 所示。



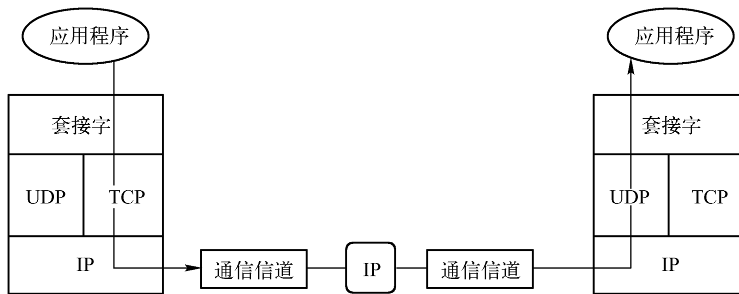


图 4-23　Socket 通信原理图

从图中进一步看出， 应用程序可以使用 UDP 或者 TCP 协议这两种套接字进行通信。当客户端和服务器端的协议相对应时，客户端使用 TCP，服务器端也使用 TCP。TCP 协议的通信过程是：首先连接接收方，然后发送数据，可以保证成功率，但速度相对较慢。UDP 协议的通信过程是：把数据打包成数据包，然后直接发送对应的 IP 地址，这种通信方式速度快，但是不保证成功率，并且数据大小有限。下面重点介绍 Socket 这两种方式进行通信所涉及的类和方法。

（1）使用基于 TCP 协议的 Socket 通信

基于 TCP 协议的 Socket 通信是指一个客户端要发起一次通信，首先必须知道运行服务器端的主机 IP 地址，然后由网络识别出目标地址，将客户端发送的信息传递到正确的主机上。在 Java 中，地址可以由一个字符串来定义，这个字符串可以使用数字型的地址（比如 192.168.1.1）也可以是主机名（example.com）这样的地址。表 4-11 列出了基于 TCP 协议操作 Socket 的 API。

表 4-11　基于 TCP 协议操作 Socket 的 API

类	功 能 说 明
InetAddress	在 Java 当中，InetAddress 类代表了一个网络目标地址，包括主机名和数字类型的地址信息
ServerSocket	这个类是实现了一个服务器端的 Socket，利用这个类可以监听来自网络的请求。 a) 创建 ServerSocket 的方法： ServerSocket(Int localPort) ServerSocket(int localport，int queueLimit) ServerSocket(int localport，int queueLimit，InetAddress localAddr) 创建一个 ServerSocket 必须指定一个端口，以便客户端能够向该端口号发送连接请求。端口的有效范围是 0-65535 b) ServerSocket 操作 Socket accept() void close() accept()方法为下一个传入的连接请求创建 Socket 实例，并将已成功连接的 Socket 实例返回给服务器套接字，如果没有连接请求，accept()方法将阻塞等待 close 方法用于关闭套接字
Socket	这个类实现了如何与客户端连接与通信 a) 创建 Socket 的方法： Socket（InetAddress remoteAddress，int remotePort） 利用 Socket 的构造函数，可以创建一个 TCP 套接字后，先连接到指定的远程地址和端口号 b) 操作 Socket 的方法： InputStream getInputStream()　　// 输入流 OutputStream getOutputStream()　　// 输出流 void close()　　//关闭方法



TCP-Socket 操作原理如图 4-24 所示。

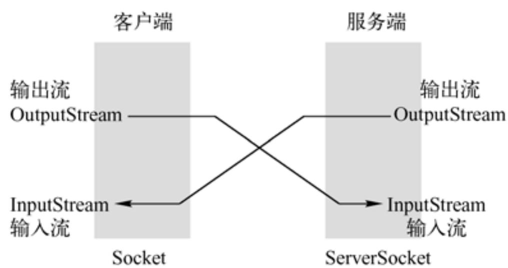


图 4-24　TCP-Socket 操作原理图

（2）使用基于 UDP 的 Socket

要使用基于 UDP 的 Socket，必须先了解 Socket 通信相关类，见表 4-12。

表 4-12　基于 UDP 的 Socket 通信相关类

类	功 能 说 明
DatagramPacket	数据报包用来实现无连接包投递服务。每条报文仅根据该包中包含的信息从一台机器路由到另一台机器这种方式可能选择不同的路由，也可能按不同的顺序到达，不对包投递的内容做出保证。具体过程如下： （1）创建 DatagramPacket DatagramSocket(byte [] data，int offset，int length，InetAddress remoteAddr，int remotePort) //该构造函数创建一个数据报文对象，数据包含在第一个参数当中 （2）创建 DatagramSocket 创建 DatagramSocket(int localPort)　//该构造函数将创建一个 UDP 套接字 （3）DatagramSocket：发送和接受 void send(DatagramPacket packet) void receive(DatagramPacket packet) send()方法用来发送 DatagramPacket 实例。一旦创建连接，数据报将发送到该套接字所连接的地址 receive()方法将阻塞等待，可接收到数据报文，并将报文中的数据复制到指定的 DatagramPacket 实例中

下面通过一个示例讲述 Socket 通信方式，该示例主要功能是模拟 Socket 通信的主要过程。

首先，介绍 Socket 通信的服务器端代码，在 eclipse 平台上建 java 项目，如代码清单 4-59、4-60 所示。

代码清单 4-59　Socket通信示例（第 4 章\PCSocketServer）服务器端代码 TCPDesktopServer.java

```
import java.io.BufferedReader;
import java.io.InputStreamReader;
import java.net.ServerSocket;
import java.net.Socket;

/**
 * socket 通信服务器端程序
 * @author
 *
 */
public class TCPDesktopServer implements Runnable {
    /*  设置本机 Ip */
    public static final String SERVERIP = "127.0.0.1";
    /*  设置端口，不要被占用，否则无法通信  */
```




```
        public static final int SERVERPORT = 4444;

        /**
         * 主程序入口
         * @param a
         */
        public static void main(String a[]) {
                Thread desktopServerThread = new Thread(new TCPDesktopServer());
                desktopServerThread.start();
        }

        public void run() {
                try {
                        /* 控制台打印 */
                        System.out.println("S: Connecting...");
                        /* 服务器端建立 socket 通信 */
                        ServerSocket serverSocket = new ServerSocket(SERVERPORT);

                        while (true) {
                                /* 服务器端接收客户端请求 */
                                Socket client = serverSocket.accept();
                                /* 控制台打印信息作为响应客户端请求 */
                                System.out.println("S: Receiving...");

                                try {
                                        /* 接收客户端传过来的信息流 */
                                        BufferedReader in = new BufferedReader(
                                                        new InputStreamReader(client.getInputStream()));
                                        String str = in.readLine();

                                        /* 控制台打印接收到的客户端信息 */
                                        System.out.println("S: Received: '" + str + "'");
                                } catch (Exception e) {
                                        System.out.println("S: Error");
                                        e.printStackTrace();
                                } finally {
                                        client.close();
                                        System.out.println("S: Done.");
                                }
                        }
                } catch (Exception e) {
                        System.out.println("S: Error");
                        e.printStackTrace();
                }
        }
}
```

代码清单 4-60　Socket通信示例（第 4 章\PCSocketServer）服务器端代码 UDPClient.java

```
import java.net.DatagramPacket;
```



```
import java.net.DatagramSocket;
import java.net.InetAddress;

public class UDPClient {
	public static void main(String[] args) {
		try {
			//首先创建一个 DatagramSocket 对象
			DatagramSocket socket = new DatagramSocket(4445);
			//创建一个 InetAddree
			InetAddress serverAddress = InetAddress.getByName("127.0.0.1");
			String str = "hello";
			byte data [] = str.getBytes();
			//创建一个 DatagramPacket 对象，
			//并指定要讲这个数据包发送到网络当中的哪个地址，以及端口号
			DatagramPacket packet
					= new DatagramPacket(data,data.length,serverAddress,1433);
			//调用 socket 对象的 send 方法，发送数据
			socket.send(packet);
		} catch (Exception e) {
			// TODO Auto-generated catch block
			e.printStackTrace();
		}
	}
}
```

其次，介绍如何在 Android 客户端的 xml 布局实现这一效果，如代码清单 4-61 所示。

代码清单 4-61　Socket通信示例Android客户端的xml布局（第 4 章\Demo_04_18）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
<TextView
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"
     />
</LinearLayout>
```

然后，介绍 Activity　java 代码如何配合布局实现 Socket 通信过程，如代码清单 4-62、4-63 所示。

代码清单 4-62　Socket通信示例（第 4 章\Demo_04_19）MainActivity.java

```
package com.chen.android;
```

```
import android.app.Activity;
import android.os.Bundle;

/**
 * socket 通信
 * @author
 */
public class MainActivity extends Activity {
        /** 当 activity 首次创建时响应调用 */
        @Override
        public void onCreate(Bundle icicle) {
                super.onCreate(icicle);
                setContentView(R.layout.main);

                /* 通过一个线程启动一个 TCPClient */
                Thread cThread = new Thread(new TCPClient());

                /* 线程启动 */
                cThread.start();

        }
}
```

代码清单 4-63　Socket通信示例（第 4 章\Demo_04_19）TCPClient.java

```
package com.chen.android;

import java.io.BufferedWriter;
import java.io.OutputStreamWriter;
import java.io.PrintWriter;
import java.net.InetAddress;
import java.net.Socket;
import android.util.Log;
/**
 * Tcp 客户端（Android 程序端）
 * @author
 *
 */
public class TCPClient implements Runnable {

        /*指定 Server 监听的端口和服务器 IP 地址（可以使用本机 IP 地址测试）*/
        public static final String SERVERIP = "192.168.1.178";
        public static final int SERVERPORT = 4444;
       public void run() {
               try {
                 /*设置一个连接的网络目标地址*/
                 InetAddress serverAddr = InetAddress.getByName(SERVERIP);
                 /*写日志*/
                 Log.d("TCP", "C: Connecting...");
                 /*实例化一个 socket 对象,4444*/
```



```
            Socket socket = new Socket(serverAddr, SERVERPORT);
            /*发送信息*/
            String message = "hello I'm from android client";
                try {
                 /*写日志，保存发送信息*/
                 Log.d("TCP", "C: Sending: '" + message + "'");
                 /*调用打印*/
                 PrintWriter out = new PrintWriter(
                                new BufferedWriter(new OutputStreamWriter(socket.getOutputStream())),
true);
                 /*控制台打印信息*/
                 out.println(message);
                 /*写日志*/
                 Log.d("TCP", "C: Sent.");
                     Log.d("TCP", "C: Done.");

              } catch(Exception e) {
                   Log.e("TCP", "S: Error", e);
                 } finally {
                   socket.close();
                 }
          } catch (Exception e) {
                Log.e("TCP", "C: Error", e);
          }
      }
  }
```

最后，需要在客户端 Android 程序中的 AndroidManifest.xml 要添加访问权限，如代码清单 4-64 所示。

代码清单 4-64　Socket通信示例（第 4 章\Demo_04_19）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.chen.android"
        android:versionCode="1"
        android:versionName="1.0">
    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-sdk android:minSdkVersion="3" />
</manifest>
```

最好在调试模式下运行，以便更清楚地弄懂整个通信过程。具体方法如下：

1）第一步，在程序中设断点，单击工具栏的调试按钮，进入调试状态，如图 4–25 所示。

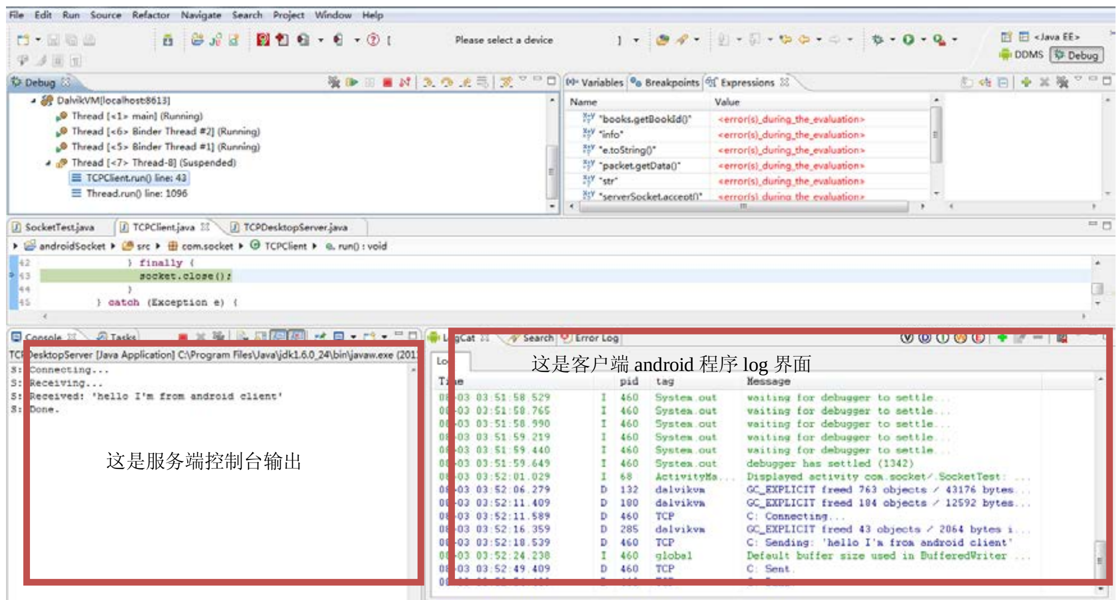


图 4–25　eclipse 开发平台客户端和服务器端调试界面

2）第二步，单步调试代码，观察服务端控制台输出与客户端 Android 程序 log（日志）文件，如图 4–26、4–27 所示。

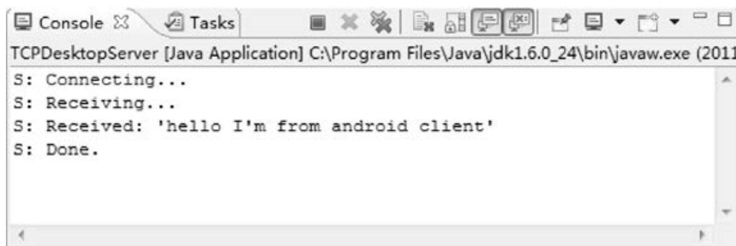
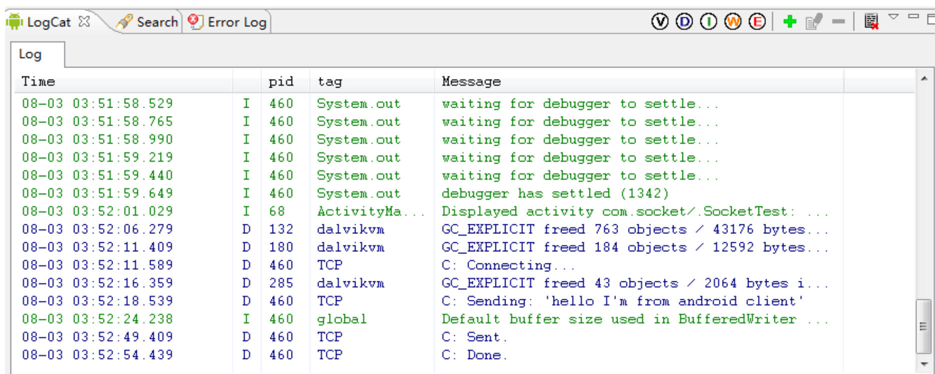


图 4–26　服务端控制台输出



Time		pid	tag	Message
08-03 03:51:58.529	I	460	System.out	waiting for debugger to settle...
08-03 03:51:58.765	I	460	System.out	waiting for debugger to settle...
08-03 03:51:58.990	I	460	System.out	waiting for debugger to settle...
08-03 03:51:59.219	I	460	System.out	waiting for debugger to settle...
08-03 03:51:59.440	I	460	System.out	waiting for debugger to settle...
08-03 03:51:59.649	I	460	System.out	debugger has settled (1342)
08-03 03:52:01.029	I	68	ActivityMa...	Displayed activity com.socket/.SocketTest: ...
08-03 03:52:06.279	D	132	dalvikvm	GC_EXPLICIT freed 763 objects / 43176 bytes...
08-03 03:52:11.409	D	180	dalvikvm	GC_EXPLICIT freed 184 objects / 12592 bytes...
08-03 03:52:11.589	D	460	TCP	C: Connecting...
08-03 03:52:16.359	D	285	dalvikvm	GC_EXPLICIT freed 43 objects / 2064 bytes i...
08-03 03:52:18.539	D	460	TCP	C: Sending: 'hello I'm from android client'
08-03 03:52:24.238	I	460	global	Default buffer size used in BufferedWriter ...
08-03 03:52:49.409	D	460	TCP	C: Sent.
08-03 03:52:54.439	D	460	TCP	C: Done.

图 4–27　客户端 Android 程序 log（日志）文件



4.3　登录功能实现

通过对本章基础知识及重点知识的讲解，相信读者实现订票系统的登录功能并不困难。需要注意的是，从本章开始要涉及与数据后台的交互，在这里将进一步讲解 Android 如何和后台进行通信，如何编写 ASP 后台代码，如何组装数据便于 Android 前端程序的解析等知识点。

4.3.1　登录界面View实现

根据第 2 章手机订票系统实现效果图（图 2-3）——登录界面的 View 实现，本示例采用 LinearLayout 线性布局中嵌套 LinearLayout 实现画面布局，读者可以采用其他布局来实现同样的效果。为了让界面更友好和美观，本例设置用户名和密码编辑框的 android:hint 属性，以便更友好地提示用户输入。详见代码清单 4-65。

代码清单 4-65　登录界面View实现（第 4 章\GanaSky）login.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:id="@+id/fligtLisy_items"
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:orientation="vertical"
     android:background="@drawable/buttonbgon">
     <!-- 第一行 -->
      <LinearLayout
         android:orientation="horizontal"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
           android:paddingTop="15dip">
         <TextView
                android:id="@+id/widget28"
                android:layout_width="wrap_content"
                 android:layout_height="wrap_content"
                 android:text="登录名：">
         </TextView>
            <EditText
                android:id="@+id/et_UserName"
                android:layout_width="fill_parent"
                 android:layout_height="wrap_content"
                 android:textSize="11sp"
                 android:hint="请输入用户名">
            </EditText>

      </LinearLayout>
         <!-- 第二行 -->
      <LinearLayout
         android:orientation="horizontal"
```




```
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1.0">
    <TextView
            android:id="@+id/widget30"
            android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:text="密码     ：">
    </TextView>
      <EditText
            android:id="@+id/et_UserPwd"
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:textSize="11sp"
             android:password="true"
             android:hint="请输入密码">
      </EditText>
  </LinearLayout>

    <!-- 第三行 -->
  <LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1.0"
    android:gravity="center_horizontal">
    <RadioGroup
            android:id="@+id/RadioGroup01"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:orientation="horizontal">
             <RadioButton
                  android:id="@+id/RadioButtonOfSave"
                   android:layout_width="wrap_content"
                   android:layout_height="wrap_content"
                   android:text="保存用户名"  >
             </RadioButton>
             <RadioButton
                  android:id="@+id/RadioButtonOfNotSave"
                   android:layout_width="wrap_content"
                   android:layout_height="wrap_content"
                   android:layout_marginLeft="5dip"
                   android:text="不保存用户名" >
             </RadioButton>
      </RadioGroup>
  </LinearLayout>
    <!-- 第四行 -->
  <LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
```



```
        android:layout_height="wrap_content"
        android:layout_weight="1.0"
        android:gravity="center_horizontal">
        <Button
                android:id="@+id/btn_Ok"
                android:layout_width="80dip"
                 android:layout_height="40dip"
                 android:text="确定"
                 >
      </Button>
      <Button
                android:id="@+id/btn_Cancel"
                android:layout_width="80dip"
                 android:layout_height="40dip"
                 android:layout_marginLeft="15dip"
                 android:text="取消"
                 >
          </Button>
      </LinearLayout>
        <!-- 第五行 -->
      <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.0"
        android:gravity="right"
        android:layout_marginRight="5dip">
        <Button
                android:id="@+id/btn_Register"
                android:layout_width="80dip"
                 android:layout_height="40dip"
                 android:text="免费注册"
                 >
        </Button>
      </LinearLayout>

</LinearLayout>
```

从登录界面开始，要涉及如何与后台通信的过程，这就要获取链接地址。为了便于管理，我们集中写在 strings.xml 文件中。详见代码清单 4-66。

代码清单 4-66　订票系统（第 7 章\Demo_07_03）strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
      <string name="hello">单击每个按钮，看看状态栏的变化！</string>
      <string name="app_Iconname">订票系统</string>
      <string name="app_ganaSkyMain">欢迎使用订票系统...</string>
      <string name="app_nameLogin">登录</string>
      <string name="app_nameRegister">注册</string>
       <string name="app_nameTicketMain">机票查询</string>
```

```
        <string name="app_nameCitySelect">城市选择</string>
        <string name="app_nameSelectFlight">选择航班</string>
         <string name="app_nameDetailFlight">航班详情</string>
         <string name="app_nameOrerTicket">订票</string>

       <!--与后台交互网址    -->
        <!--  登录  -->
       <string name="LoginaspURL">http://220.165.15.216:7777/MobilePage/LoginHtmlPage.aspx</string>

        <!--  注册  -->
       <string name="RegisterURL">http://220.165.15.216:7777/MobilePage/RegisterPage.aspx</string>

        <!--  匹配城市  -->
       <string name="CitySelectURL">http://220.165.15.216:7777/MobilePage/CityQueryPage.aspx</string>

         <!--  查询机票 ticketSearchUrl -->
       <string name="TicketSearchUrl">http://220.165.15.216:7777/MobilePage/SearchTicketPage.aspx</string>

         <!--  订票  -->
      <string name="OrderTicketURL">http://220.165.15.216:7777/MobilePage/OrderTicket.aspx</string>

</resources>
```

4.3.2 登录功能Model用户信息类实现

登录功能涉及要保存用户信息，因此有必要建一个 Model 类来保存用户信息，这需要在该类中对用户信息的相关属性进行封装。详见代码清单 4-67。

代码清单 4-67　登录界面Model实现（第 4 章\GanaSkyForLogin）UserInfo.java

```
package com.ganasky.model;

/**
 * 用户信息类
 * @author
 *
 */
public   class UserInfo {
        public final static String USERNAME="userName";
        public final static String USERREALNAME="userRealName";
        public final static String CHECKLEVEL="checklevel";

        private String userName;
        private String userPwd;
        private String userRealName;
        private String checkLevel;
        private String userMobilePhone;
        private String userId;

        private String userEmail;
```



```
        public String getUserRealName() {
            return userRealName;
        }
        public void setUserRealName(String userRealName) {
            this.userRealName = userRealName;
        }
        public String getCheckLevel() {
            return checkLevel;
        }
        public void setCheckLevel(String checkLevel) {
            this.checkLevel = checkLevel;
        }


        public String getUserName() {
            return userName;
        }
        public void setUserName(String userName) {
            this.userName = userName;
        }
        public String getUserPwd() {
            return userPwd;
        }
        public void setUserPwd(String userPwd) {
            this.userPwd = userPwd;
        }
        public String getUserMobilePhone() {
            return userMobilePhone;
        }
        public void setUserMobilePhone(String userMobilePhone) {
            this.userMobilePhone = userMobilePhone;
        }
        public String getUserEmail() {
            return userEmail;
        }
        public void setUserEmail(String userEmail) {
            this.userEmail = userEmail;
        }
        public String getUserId() {
            return userId;
        }
        public void setUserId(String userId) {
            this.userId = userId;
        }

    }
```

4.3.3　登录功能Control实现

登录功能的 Control 实现，就是登录功能的 java Activity 实现。见代码清单 4-68。在实

现过程中需要注意的有以下几点：

1）用户登录名的保存。用户登录名的信息量相对较少，因此我们使用 SharedPreferences 保存用户信息，这涉及本章知识重点中的 SharedPreferences 的使用。

2）对用户输入信息的验证。为了让订票系统程序更加健壮，减轻后台通信的压力，要对输入的用户名和密码进行不为空的验证。

3）如何与 ASP 后台进行通信，在重点剖析章节中也已经详细讲解过，这里进一步巩固这一知识点。

4）解析登录后返回的用户信息数据，本项目采用 DOM 方式解析数据，这种方式较直观易懂，便于初学者掌握 XML 解析的过程。

代码清单 4-68　登录功能Control实现（第 4 章\GanaSkyForLogin）LoginApp.java（详见本书源代码）

4.3.4　登录功能Help实现

在 Help 包中新建 ConnUrlHelper 类，用来保存与数据后台进行通信的几种方法，这里主要定义了 HttpURLConnection 和 HttpClient 这两类通信方式的多种方法，因为不是点对点的程序，不必考虑使用 Socket 通信。详见代码清单 4-69。

代码清单 4-69　登录界面Help实现（第 4 章\GanaSkyForLogin）ConnUrlHelper.java（详见本书源代码）

在代码中使用到了 Log.e，这个方法的主要功能是为应用程序写日志，便于捕获异常。在程序调试状态下， DDMS 透视图中可以看到相应的日志文件，如图 4-28 所示。

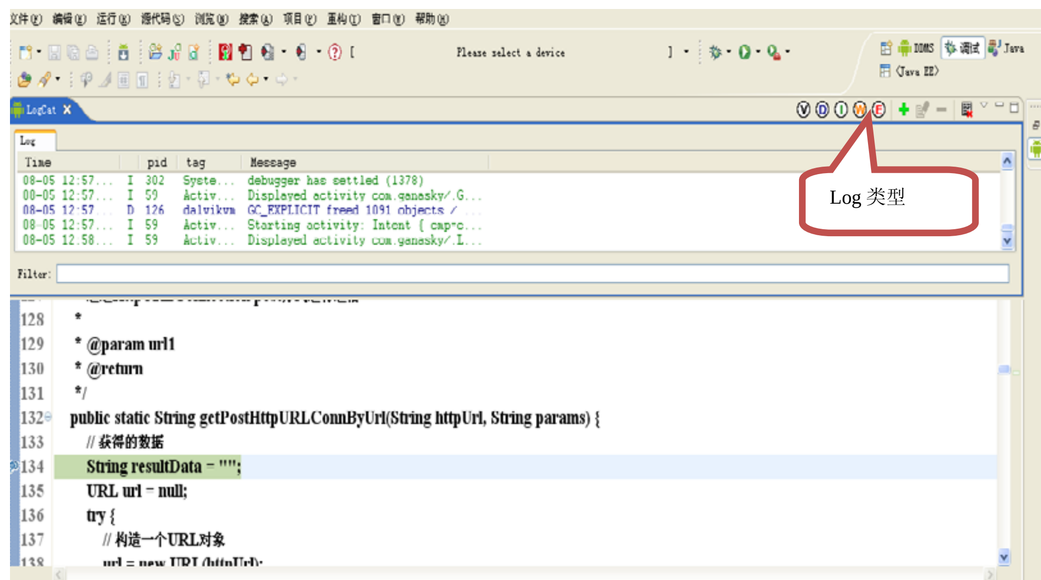


图 4-28　Log 调试界面

4.3.5　登录后台ASP实现

从这一章节开始，将涉及手机机票系统的 ASP 后台技术实现，在这里主要是使用 C#语



言开发的 ASP 网站后台，如何在 Visual Studio 上建 ASP 网站，不是本书所讲述的重点。本书之所以要强调 ASP 后台开发，是因为在本项目开发中所书写的 ASP 界面与一般 Web 网站的 ASP 界面不同，只有两层（逻辑层 BLL 和数据库层 DAL）。本书主要是想介绍 ASP 后台的开发模式，如何使用 XML 技术，如何组合数据与前台 Android 程序进行增、删、改、查的交互操作。在书中会给出 ASP 逻辑层的代码（BLL 层），如何与数据库后台进行连接不是本书讨论的重点，因此代码片段不能单独运行。没有 C#开发经验的读者可以阅读其他方面的书籍。如果读者急于查看订票系统的运行结果，可以直接调用本书的网站接口，本书发布一个公共网站，供读者调用（仅供学习）。因为目前国内有多家公司做订票接口，也可以不编写 ASP 后台代码，直接调用其他公司（如 51book）的 Web 接口，从而实现查票、订票、出票及退票等功能。见代码清单 4-70。

代码清单 4-70　登录后台ASP实现（第 4 章\GanaSkyForLogin）LoginHtmlPage.aspx.cs（详见本书源代码）

登录功能执行 ASP 后台代码后返回的 xml 数据，如代码清单 4-71 所示。

代码清单 4-71　登录功能执行ASP后台代码后返回的xml数据（第 4 章\GanaSkyForLogin\assets\data）userInfo.xml

```
<?xml version="1.0" encoding="utf-8"?>
<UsersInfo>
  <Users>
    <Id>4</Id>
    <Company_Id>1</Company_Id>
    <UserCllass>资金管理中心经理</UserCllass>
    <UserName>付显蓉</UserName>
    <LoginName>000377</LoginName>
    <Password>e10adc3949ba59abbe56e057f20f883e</Password>
    <MobileNo>15987100000</MobileNo>
    <E_Mail>KMG239@126.COM</E_Mail>
    <PhoneNo>0871-3100000</PhoneNo>
    <City>昆明</City>
    <Electrograph>0871-3199566</Electrograph>
    <Defray>18908714585</Defray>
    <ComTencentpaycenter>468894917</ComTencentpaycenter>
    <Msnqq>468894917</Msnqq>
    <ParentID></ParentID>
    <Enabled>True</Enabled>
    <Permissions></Permissions>
    <Version>"AAAAAAAB9Bc="</Version>
    <Company>GanaSky.Database.Company</Company>
  </Users>
</UsersInfo>
```

第 5 章　注册功能实现

本章是对第 4 章基础控件的巩固，没有涉及新的控件学习，主要是想强调一种循环渐进，温故而知新的学习过程。本章涉及的知识重点有两个：

1）如何实现不同界面的传值，这是对第 3 章页面切换的进一步深化与学习。

2）注册页面涉及多种用户信息的验证，本章将重点讲述如何在 Android 程序开发中使用正则表达式（Regular Expression）对用户输入的信息进行验证。

5.1　重点剖析

重点剖析节介绍需要重点掌握的 Android 编程方法与编程思路，本节主要介绍页面传值的技巧和使用正则表达式（Regular Expression）对用户输入的信息进行验证的内容。

5.1.1　Activity页面传值

如何实现各个 Activity 页面的传值？可以使用 intent.putExtra("key", "value")方法传递参数。在页面传值过程中，当启动一个新的页面时，不能使用调用者的 finish()方法（如：MainActivity.this.finish()），否则无法传值，而应该使用调用者的 startActivity()方法（如 MainActivity.this.startActivity(intent)）。下面通过一个示例，讲述 Activity 页面传值的用法，代码运行结果如图 5-1 和图 5-2 所示。

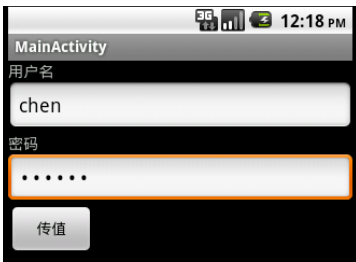


图 5-1　Activity 页面传值示例运行结果（1）

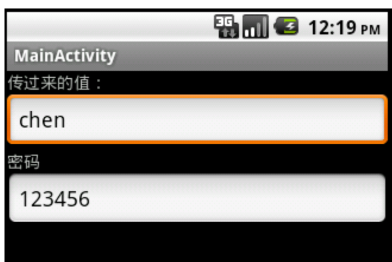


图 5-2　Activity 页面传值示例运行结果（2）

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 5-1、5-2 所示。

代码清单 5-1　第 5 章Activity页面传值示例（第 5 章\Demo_05_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```



```
            android:layout_height="fill_parent"
            >
        <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="用户名"
            />
        <EditText
               android:id="@+id/txtUserName"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:maxLength = "20"
               />
        <TextView
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="密码"
                />
        <EditText
               android:id="@+id/txtPass"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:maxLength = "20"
               android:password = "true"
               />
         <LinearLayout
               xmlns:android="http://schemas.android.com/apk/res/android"
               android:orientation="horizontal"
               android:layout_width="fill_parent"
               android:layout_height="fill_parent"
                >
         <Button
               android:id="@+id/btnSend"
               android:layout_width="wrap_content"
               android:layout_height="wrap_content"
               android:text="传值"
               android:gravity = "center"
               android:width = "80px"
               />
         </LinearLayout>
    </LinearLayout>
```

代码清单 5-2　第 5 章Activity页面传值示例（第 5 章\Demo_05_01）　register.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">
     <TextView
```

```
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="传过来的值：" />
        <EditText
            android:id="@+id/ReUserName"
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:maxLength="20" />
        <TextView
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="密码" />
        <EditText
            android:id="@+id/RePass"
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:maxLength="20"
             android:password="true" />
</LinearLayout>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现页面切换及传值这一功能，如代码清单 5-3、5-4 所示。

代码清单 5-3 第 5 章Activity页面传值示例（第 5 章\Demo_05_01）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.content.Intent;
/**
  * Demo_05_01Activity 页面切换及传值
 * @author
  *
  */
public class MainActivity extends Activity {

        private Button btnsend;
        private EditText username;
        private EditText pass;

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                 btnsend = (Button) this.findViewById(R.id.btnSend);
                 username = (EditText) this
```



```
                        .findViewById(R.id.txtUserName);
                pass = (EditText) this.findViewById(R.id.txtPass);

                btnsend.setOnClickListener(login);
        }
        /**
         * 登录
         */
        Button.OnClickListener login=new Button.OnClickListener()
        {
                @Override
                public void onClick(View v) {
                        String un = username.getText().toString().trim();
                        String ps = pass.getText().toString().trim();
                        if (!"".equals(un) && !"".equals(ps)) {
                                /* 新建一个 Intent 对象 */
                                Intent intent = new Intent();
                                if (un != null) {
                                        // 设置传递的参数
                                        intent.putExtra("username", un);
                                        intent.putExtra("userPwd", ps);
                                }
                                /* 指定 intent 要启动的类 */
                                intent.setClass(MainActivity.this, Register.class);
                                // 启动 intent 的 Activity
                                MainActivity.this.startActivity(intent);

                        } else {
                        }
                }

        };

}
```

代码清单 5-4　第 5 章Activity页面传值示例（第 4 章\Demo_05_01）Register.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;

public class Register extends Activity {
   private String username;
   private String userPwd;

 /** 当 activity 首次创建时响应调用 */
     @Override
     public void onCreate(Bundle savedInstanceState) {
```

```
        super.onCreate(savedInstanceState);
            setContentView(R.layout.register);
             /*接收页面传过来的值*/
            Intent intent = getIntent();
            username=intent.getStringExtra("username");
            userPwd=intent.getStringExtra("userPwd");

            /*给控件赋值*/
            TextView tv_userName=(TextView)this.findViewById(R.id.ReUserName);
            TextView tv_userPwd=(TextView)this.findViewById(R.id.RePass);
            tv_userName.setText(username) ;
            tv_userPwd.setText(userPwd) ;

    }
}
```

再者，在 AndroidManifest.xml 中，需要添加相应的 Activity，如代码清单 5-5 所示。

代码清单 5-5　第 5 章Activity页面传值示例（第 4 章\Demo_05_01）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.chen.android"
        android:versionCode="1"
        android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon"
                                        android:label="@string/app_name">
        <activity android:name=".MainActivity"
                             android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
         <activity android:name="Register"></activity>
    </application>
</manifest>
```

5.1.2　使用正则表达式进行信息验证

对用户输入的信息进行验证，就是尽量减少后台代码的压力，特别是对邮箱、数字等用户输入的信息进行验证，可以提高系统的效率以及系统的健壮性。对用户输入的信息进行验证，首先验证用户输入的信息是否为空，不为空的情况下再使用正则表达式对信息进行进一步的验证。

所谓正则表达式就是使用一个“字符串”来描述一个特征，然后去验证另一个“字符串”是否符合这个特征。比如表达式“ac+”描述的特征是“一个 'a' 和 任意个 'c' ”，那么



'ac'， 'acc'， 'acccccccccc'都符合这个特征。使用正则表达式对用户输入的信息进行验证的好处是：

1）验证字符串是否符合指定特征，比如验证用户输入的邮箱地址是否是合法的邮件地址。

2）用于查找字符串，可以从一个长的文本中查找符合指定特征的字符串，比查找固定字符串更加灵活方便。

3）用于替换，使用正则表达式进行字符串的替换比普通的替换更强大。

下面通过一个示例讲述如何使用正则表达式对用户输入的信息进行验证，示例代码运行结果如图 5-3 至图 5-5 所示。



图 5-3　使用正则表达式进行用户信息验证示例（1）——初始化



图 5-4　使用正则表达式进行用户信息验证示例（2）——填写用户信息



图 5-5　使用正则表达式进行用户信息验证示例（3）——验证成功

首先，介绍如何通过 xml 布局实现正则表达式用户信息验证，如代码清单 5-6 所示。

代码清单 5-6　使用正则表达式进行用户信息验证示例（第 5 章\Demo_05_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/fligtLisy_items"
     xmlns:android="http://schemas.android.com/apk/res/android"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical">
      <!-- 第一行 -->
       <LinearLayout
          android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content">
          <TextView
                  android:id="@+id/widget34"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="登录名　："> 
```

```
          </TextView>
            <EditText
                  android:id="@+id/ed_registeruserName"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:textSize="11sp"
                  android:hint="请输入用户名">
            </EditText>
        </LinearLayout>
        <!-- 第二行 -->
        <LinearLayout
          android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1.0"
          android:paddingTop="5dip">
          <TextView
                  android:id="@+id/widget37"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="密码      ：">
          </TextView>
            <EditText
                  android:id="@+id/ed_registerPwd"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:textSize="11sp"
                  android:hint="建议设置6位以上的密码"
                  android:password="true">
            </EditText>
        </LinearLayout>
        <!-- 第三行 -->
        <LinearLayout
          android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_weight="1.0">
          <TextView
                  android:id="@+id/widget39"
                  android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:text="确认密码:">
          </TextView>
            <EditText
                  android:id="@+id/ed_registerRePwd"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:textSize="11sp"
                  android:hint="请再次输入密码"
                  android:password="true">
```



```
        </EditText>
    </LinearLayout>
    <!-- 第四行 -->
    <LinearLayout
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_weight="1.0">
      <TextView
            android:id="@+id/widget41"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="手机号码:">
      </TextView>
        <EditText
            android:id="@+id/ed_registerUserMoblePhone"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="11sp"
            android:numeric="integer"
            android:hint="请填写您的手机号码，以方便您预订机票">
        </EditText>

    </LinearLayout>
    <!-- 第五行 -->
    <LinearLayout
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:layout_weight="1.0">
      <TextView
            android:id="@+id/widget43"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="电子邮箱:">
      </TextView>
      <EditText
            android:id="@+id/ed_registerUserEmail"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="11sp"
            android:hint="请填写您的邮箱，以便联系">
        </EditText>

    </LinearLayout>
    <!-- 第六行 -->
    <LinearLayout
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
```




```
        android:layout_weight="1.0"
        android:gravity="center_horizontal">
        <Button
            android:id="@+id/btn_Register"
            android:layout_width="80dip"
            android:layout_height="40dip"
            android:text="注册">
        </Button>
        <Button
            android:id="@+id/btn_RegisterCancel"
            android:layout_width="80dip"
            android:layout_height="40dip"
            android:layout_marginLeft="10dip"
            android:text="取消" >
        </Button>

    </LinearLayout>
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一功能，如代码清单 5-7 所示。

代码清单 5-7　使用正则表达式进行用户信息验证（第 5 章\Demo_05_02）MainActivity.java

```
package com.chen.android;

import java.util.regex.Matcher;
import java.util.regex.Pattern;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Gravity;
import android.view.MotionEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;

/**
 * Demo_05_02，如何实现验证
 * @author
 */
public class MainActivity extends Activity {
    /**当 activity 首次创建时响应调用*/

    private EditText ed_userName;
    private EditText ed_userPwd;
    private EditText ed_userRePwd;
    private EditText ed_userMobilePhone;
    private EditText ed_userEmail;
    private Button btn_Register;
```



```
 private Button btn_RegisterCancel;
@Override
public void onCreate(Bundle savedInstanceState) {
      super.onCreate(savedInstanceState);
      setContentView(R.layout.main);

      ed_userName = (EditText) findViewById(R.id.ed_registeruserName);
        ed_userPwd = (EditText) findViewById(R.id.ed_registerPwd);
        ed_userRePwd = (EditText) findViewById(R.id.ed_registerRePwd);
        ed_userMobilePhone = (EditText) findViewById(R.id.ed_registerUserMoblePhone);
        ed_userEmail = (EditText) findViewById(R.id.ed_registerUserEmail);
        btn_Register = (Button) findViewById(R.id.btn_Register);
        btn_RegisterCancel=(Button) findViewById(R.id.btn_RegisterCancel);

        // 设置监听
        btn_Register.setOnTouchListener(btnRegisterTouch);
        btn_RegisterCancel.setOnTouchListener(btnRegisterCancelTouch);

}

Button.OnTouchListener btnRegisterTouch = new Button.OnTouchListener() {
        @Override
        public boolean onTouch(View v, MotionEvent event) {
               // 设置哪些触屏事件可用
               int iAction = event.getAction();
               if (iAction == MotionEvent.ACTION_CANCEL
                             || iAction == MotionEvent.ACTION_DOWN
                             || iAction == MotionEvent.ACTION_MOVE) {
                      return false;
               }
               // 客户端判断
               if (checkUserInfo()) {
                      DisplayToast("信息验证成功！");
               }
               return true;
        }
 };

 /**
  * 触屏取消登录的方法
  */
 Button.OnTouchListener btnRegisterCancelTouch = new Button.OnTouchListener() {

        @Override
        public boolean onTouch(View v, MotionEvent event) {
               // 设置哪些触屏事件可用
               int iAction = event.getAction();
               if (iAction == MotionEvent.ACTION_CANCEL
                             || iAction == MotionEvent.ACTION_DOWN
                             || iAction == MotionEvent.ACTION_MOVE) {
                      return false;
```




```
                }
                /*退出*/
                exitApp();
                return true;
            }

    };

    /**
     * 验证用户信息
     */
    private boolean checkUserInfo() {
            if (ed_userName.getText() != null && ed_userName.getText().length() > 0) {
            } else {
                DisplayToast("用户名不能为空！");
                return false;
            }
            if (ed_userName.getText().length() < 6) {
                DisplayToast("用户名太短！");
                return false;
            }

            if (ed_userPwd.getText() != null && ed_userPwd.getText().length() > 0) {

            } else {
                DisplayToast("密码不能为空！");
                return false;
            }
            if (ed_userPwd.getText().length() < 6) {
                DisplayToast("密码太短！");
                return false;
            }

            if (ed_userRePwd.getText() != null
                        && ed_userRePwd.getText().length() > 0
                        && ed_userPwd.getText().toString()
                                    .equals(ed_userRePwd.getText().toString())) {

            } else {
                DisplayToast("两次密码不一致！");
                return false;
            }

            if (ed_userMobilePhone.getText() != null
                        && ed_userMobilePhone.getText().length() > 0) {

            } else {
                DisplayToast("请填写电话号码！");
                return false;
            }
            if (isNumeric(ed_userMobilePhone.getText().toString())) {
```



```
        } else {
            DisplayToast("手机只能为数字！");
            return false;

        }
        if ( ed_userMobilePhone.getText().length()==11) {

        } else {
            DisplayToast("手机号码必须是十一位！");
            return false;
        }

        if (ed_userEmail.getText() != null
                && ed_userEmail.getText().length() > 0) {
        } else {
            DisplayToast("请填写邮箱！");
            return false;
        }

        if (getEmail(ed_userEmail.getText().toString())) {

        } else {
            DisplayToast("邮箱格式不正确！");
            return false;
        }
        return true;

    }

    /**
     * 数字验证
     *
     * @param str
     * @return
     */
    private boolean isNumeric(String str) {
        //定义正则表达式
        String regNumeric="[0-9]*";
        Pattern pattern = Pattern.compile(regNumeric);
        return pattern.matcher(str).matches();
    }

    /**
     * 邮箱验证
     * @param strEmail
     * @return
     */
    private boolean getEmail(String strEmail) {
        /*定义正则表达式*/
        String regEmail="\\w+([-+.]\\w+)*@\\w+([-.]\\w+)*\\.\\w+([-.]\\w+)*";
```




```
            Pattern p = Pattern
                    .compile(regEmail);
            Matcher m = p.matcher(strEmail);
            return m.find();
        }

        /*  显示 Toast */
        public void DisplayToast(String str) {
            Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
            //  设置 toast 显示的位置
            toast.setGravity(Gravity.TOP, 0, 220);
            //  显示该 toast
            toast.show();
        }

        /**
         * Android 2.2 退出
         */
        private void exitApp() {
            //  先溢出用户信息
            Intent startMain = new Intent(Intent.ACTION_MAIN);
            startMain.addCategory(Intent.CATEGORY_HOME);
            startMain.setFlags(Intent.FLAG_ACTIVITY_NEW_TASK);
            startActivity(startMain);
            System.exit(0);

        }
    }
```

5.2 注册功能实现

注册功能是对第 4 章登录界面所涉及的知识点的巩固，相对简单。需要注意的是，在注册过程中可能使用中文，需要使用 UTF-8 或者简体中文（gb2312）编码格式处理中文乱码的问题，且 Android 程序的编码格式要和 ASP 程序的编码格式保持一致。

5.2.1 注册界面View实现

根据第 2 章手机订票系统注册功能实现效果图（图 2-4）所示，本例也是采用 LinearLayout 线性布局中嵌套 LinearLayout 实现页面效果。详见代码清单 5-8。

代码清单 5-8 注册界面View实现（第 5 章\ GanaSkyFoRegister）register.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/linearLayout_Register"
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:orientation="vertical"
```



```
android:background="@drawable/buttonbgon">
<!-- 第一行 -->
 <LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView
            android:id="@+id/widget34"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="登录名　：">
    </TextView>
      <EditText
            android:id="@+id/ed_registeruserName"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="11sp"
            android:hint="请输入用户名">
      </EditText>
 </LinearLayout>
 <!-- 第二行 -->
 <LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1.0"
    android:paddingTop="5dip">
    <TextView
            android:id="@+id/widget37"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:text="密码　　：">
    </TextView>
      <EditText
            android:id="@+id/ed_registerPwd"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:textSize="11sp"
            android:hint="建议设置6位以上的密码"
            android:password="true">
      </EditText>
 </LinearLayout>
 <!-- 第三行 -->
 <LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:layout_weight="1.0">
    <TextView
            android:id="@+id/widget39"
```




```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="确认密码:">
        </TextView>
          <EditText
                android:id="@+id/ed_registerRePwd"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textSize="11sp"
                android:hint="请再次输入密码"
                android:password="true">
          </EditText>
    </LinearLayout>
    <!-- 第四行 -->
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.0">
        <TextView
                android:id="@+id/widget41"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="手机号码:">
        </TextView>
          <EditText
                android:id="@+id/ed_registerUserMoblePhone"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:textSize="11sp"
                android:numeric="integer"
                android:hint="请填写您的手机号码，以方便您预订机票">
          </EditText>

    </LinearLayout>
    <!-- 第五行 -->
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.0">
        <TextView
                android:id="@+id/widget43"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="电子邮箱:">
        </TextView>
        <EditText
                android:id="@+id/ed_registerUserEmail"
                android:layout_width="fill_parent"
```



```
                android:layout_height="wrap_content"
                android:textSize="11sp"
                android:hint="请填写您的邮箱，以便联系">
            </EditText>

        </LinearLayout>
        <!-- 第六行 -->
        <LinearLayout
            android:orientation="horizontal"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_weight="1.0"
            android:gravity="center_horizontal">
            <Button
                android:id="@+id/btn_Register"
                android:layout_width="80dip"
                android:layout_height="40dip"
                android:text="注册"
                >
            </Button>
            <Button
               android:id="@+id/btn_RegisterCancel"
               android:layout_width="80dip"
                android:layout_height="40dip"
                android:layout_marginLeft="10dip"
                android:text="取消"
                >
            </Button>

        </LinearLayout>
    </LinearLayout>
```

5.2.2　注册功能Control实现

注册功能的 Control 实现，就是注册功能的 java Activity 实现，如代码清单 5-9 所示。在实现过程中需要注意的有以下几点：

1）对用户输入信息的验证。为了使平台健壮，减轻后台的压力，要对输入的所有信息进行验证，如何对邮箱、数字使用正则表达式进行验证，在重点剖析中已涉及。

2）如何拼接用户信息并被后台 ASP 程序所解析，这是通信的关键。本书提倡采用 XML 方式构造数据源，然后封装成相应的参数，使用 GET 或者 POST 方法与 ASP 数据后台进行通信。

3）如何与后台进行通信，主要是使用 Help 包中的 ConnUrlHelper 类所定义的这几个公用方法。需要注意的是，在组装数据的时候，要考虑中文传输过程中不乱码的问题。

代码清单 5-9　注册功能 Control 实现（第 5 章\ GanaSkyFoRegister） RegisterApp.java（详见本书源代码）

5.2.3 注册后台ASP实现

注册功能执行 ASP 后台代码后返回的 XML 格式的数据有两种，一种是注册成功，如代码清单 5-10、5-11 所示。101 表示注册成功，生成的 ID。另一种是注册失败，如代码清单 5-12 所示。

代码清单 5-10 注册功能ASP实现（第 5 章\GanaSkyFoRegister） RegisterPage.aspx.cs（详见本书源代码）

代码清单 5-11 执行ASP注册成功后返回的XML格式的数据

```
<?xml version="1.0" encoding="utf-8"?>
<Result>
101 用户注册成功
</Result>
```

代码清单 5-12 注册功能执行ASP后台代码后返回的XML格式的数据（注册失败）

```
<?xml version="1.0" encoding="utf-8"?>
<Result>
False
</Result>
```

第 6 章　选择功能实现

本章要实现城市选择功能。涉及的基础控件有：自动匹配（AutoCompleteTextView）控件、列表（ListView）控件的使用。

在本章的重点知识部分将进一步学习数据源适配器，主要是学习如何配置 ListView 的数据源对象，实现 ListView 显示多种形态的内容信息，如显示多行、游标类型的子项。这一知识点是学习 Android 开发的重点，形式复杂多变，因此读者需要不断的巩固和深入。常用的 ListView 控件数据源形式有：

1）ListView 数据源组合形式：ListAdapter+ HashMap+ ArrayList，这是 ListView 常用的数据源组装形式。

2）ListView 另一种数据源形式：SimpleCursorAdapter。

6.1　基础控件讲解

本节涉及选择功能中使用的自动匹配（AutoCompleteTextView）控件、列表（ListView）控件，这些控件也是其他项目中使用的高频控件。

6.1.1　自动匹配

自动匹配（AutoCompleteTextView）实现动态匹配输入的文本内容。它实现的效果如同 Google 搜索引擎，当在框中输入文本信息时，可以根据内容显示匹配的热门信息。AutoCompleteTextView 控件也是实现这样的功能，当在搜索框里输入一些字符时（默认两个字符）会自动弹出一个下拉框显示匹配的结果。自动匹配涉及的常用方法见表 6-1。

表 6-1　AutoCompleteTextView 自动匹配常用方法

AutoCompleteTextView 常用方法	功 能 说 明
setThreshold()	设置输入几个字母开始关联，默认值是 2
clearListSelection()	清除选中的列表项
dismissDropDown()	如果已存在 AutoCompleteTextView，则关闭下拉菜单
getAdapter()	获取数据源适配器
getText()	获取文本信息

下面通过一个示例，讲述 AutoCompleteTextView 自动匹配的用法，示例代码运行结果如图 6-1 所示。

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 6-1 所示。



图 6-1　AutoCompleteTextView 自动匹配示例运行结果

代码清单 6-1　自动匹配（AutoCompleteTextView）示例（第 6 章\Demo_06_01）　main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```

```
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="horizontal"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content">
       <AutoCompleteTextView
              android:id="@+id/edit"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"/>
       <Button
              android:id="@+id/btnAdd"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="添加新项"/>
</LinearLayout>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现这一效果，如代码清单 6-2 所示。

代码清单 6-2 自动匹配（AutoCompleteTextView）示例（第 6 章\Demo_06_01）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.widget.Button;
import android.widget.ArrayAdapter;
import android.widget.AutoCompleteTextView;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;

/**
 * Demo_06_01 AutoCompleteTextView 的使用
 *
 * @author
 */
public class MainActivity extends Activity {

       AutoCompleteTextView autoview;
       ArrayAdapter<String> adapter;

       // 定义数据源
       String[] city = { "shanghai", "beijing", "changsha", "shangtou", "beihai",
                     "abcd3", "changchun" };

       @Override
       protected void onCreate(Bundle savedInstanceState) {
              super.onCreate(savedInstanceState);
              /* 设置显示 main.xml 布局 */
              setContentView(R.layout.main);
              /* 初始化数据源 */
              intialView();
              /* 设置 AutoCompleteTextView 输入几个字母开始关联，默认值是 2 */
              autoview.setThreshold(1);
```



```
            /* 取得布局 main.xml 中的 button */
            Button btn = (Button) findViewById(R.id.btnAdd);

            btn.setOnClickListener(new OnClickListener() {
                public void onClick(View v) {
                    // TODO Auto-generated method stub
                    /* 取得文本内容 */
                    String string = autoview.getText().toString();
                    /* 往数据源中动态添加数据 */
                    adapter.add(string);

                }
            });
        }

        /**
         * 绑定数据
         */
        public void intialView() {
            // 关联关键字
            adapter = new ArrayAdapter<String>(this,
                    android.R.layout.simple_dropdown_item_1line, city);

            autoview = (AutoCompleteTextView) findViewById(R.id.edit);
            autoview.setWidth(200);

            // 将 adapter 适配器添加到 AutoCompleteTextView 中
            autoview.setAdapter(adapter);
        }

    }
```

6.1.2 列表视图

列表视图（ListView）是通过垂直滚动条显示列表信息的列表视图。它以列表的形式展示具体内容，并且能够根据数据的长度自适应显示。表 6-2 列出 ListView 常用 xml 属性及方法。

表 6-2 ListView 常用 xml 属性及方法

ListView 常用 xml 属性及方法	描 述
android:choiceMode	规定 ListView 所使用的选择模式。默认状态下，ListView 没有选择模式。属性值必须设置为下列常量之一： none，值为 0，表示无选择模式 singleChoice，值为 1，表示最多可以有一项被选中 multipleChoice，值为 2，表示可以多项被选中
android:divider	规定 ListView 子项之间用某个图形或颜色来分隔
android:dividerHeight	分隔符的高度。若没有指明高度，则用此分隔符固有的高度
android:entries	引用一个将使用在此 ListView 里的数组
android:footerDividersEnabled	设成 flase 时，此 ListView 将不会在页脚视图前画分隔符，此属性默认值为 true
android:headerDividersEnabled	设成 flase 时，此 ListView 将不会在页眉视图后画分隔符，此属性默认值为 true
setChoiceMode()	设置子项选择模式，可以是单选，多选等模式
setListAdapter ()	设置与此组件相关的适配器
onListItemClick ()	对 ListView 的子项进行监听，以便对子项进行具体操作

一个 ListView 的数据要正确显示在界面上，必须包括三个部分：一是 ListView 控件（用来显示数据的列表）；二是 Data（需要显示的数据）；三是需要一个绑定 Data 和 ListView 的适配器 ListAdapter。根据列表的适配器类型，列表分为三种：ArrayAdapter、SimpleAdapter 和 SimpleCursorAdapter。其中，以 ArrayAdapter 最为简单，因为是一维数组，作为 ListView 数据源只能展示一行字。SimpleCursorAdapter 可以认为是 SimpleAdapter 对数据库的简单结合，可以方便地把数据库的内容以列表的形式展示出来。下面的示例是使用 ArrayAdapter 作为 ListView 数据源对象，其他两种列表适配器将作为本章的重点章节部分的知识点进行讲解。

下面通过一个示例讲述 ListView 的用法，使用 ArrayAdapter 作为数据源，功能是使用 ListView 显示多列数据源信息。示例代码运行结果如图 6–2 和图 6–3 所示。

首先介绍如何通过 xml 布局实现这一效果，如代码清单 6–3 所示。



图 6–2　列表控件（ListView）运行结果（1）

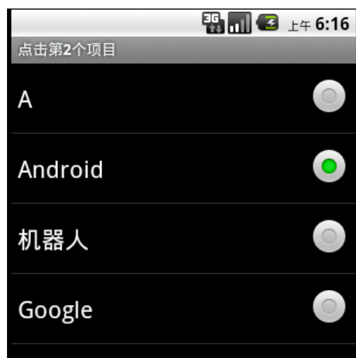


图 6–3　列表控件（ListView）运行结果（2）

代码清单 6–3　列表控件（ListView）示例（第 6 章\Demo_06_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
         android:orientation="vertical"
         android:layout_width="fill_parent"
        android:layout_height="fill_parent">

        <ListView android:id="@id/android:list"
                android:layout_width="fill_parent"
                android:layout_height="fill_parent"
                android:layout_weight="1"
         />

        <TextView android:id="@id/android:empty"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="No data"
                android:textColor="#ff0000"
         />
 </LinearLayout>
```



其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 6-4 所示。

代码清单 6-4　列表控件（ListView）示例（第 6 章\Demo_06_02）MainActivity.java

```
package com.chen.android;

import android.os.Bundle;
import android.view.View;
import android.app.ListActivity;
import android.widget.ArrayAdapter;
import android.widget.ListView;
/**
 * Demo_06_02 List 的使用
 *
 * @author
 */
public class MainActivity extends ListActivity {
        /*定义数据源*/
        private String[] mStrings = { "A", "Android", "机器人", "Google" };

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                /* 可以通过更改样式，实现不同的外观风格
                android.R.layout.simple_list_item_1  一行 text
                android.R.layout.simple_list_item_2  一行 title，一行 text
                android.R.layout.simple_list_item_single_choice  单选按钮
                android.R.layout.simple_list_item_multiple_choice  多选按钮
                android.R.layout.simple_list_item_checked checkbox */

                // //////////////////////方式一（普通方式）///////////////////////////////
                /*setListAdapter(new ArrayAdapter<String>(this,
                                android.R.layout.simple_list_item_1, mStrings));

                在 ListView 上输入字母，就会自动筛选出以此内容开头的 Item
                getListView().setTextFilterEnabled(true); */

                // //////////////////////方式二（单选模式）///////////////////////////////
                setListAdapter(new ArrayAdapter<String>(this,
                                android.R.layout.simple_list_item_single_choice, mStrings));

                final ListView listView = getListView();
                listView.setItemsCanFocus(false);
                listView.setChoiceMode(ListView.CHOICE_MODE_SINGLE); //  设置单选模式

        }

        /**
         *  重写 onListItemClick 方法
         */
        @Override
        protected void onListItemClick(ListView l, View v, int position, long id) {
                // TODO Auto-generated method stub
```




```
                super.onListItemClick(l, v, position, id);
                setTitle("单击第" + (position + 1) + "个项目");
            }
        }
```

6.2 重点剖析

重点剖析节介绍功能实现中需要重点掌握的 Android 编程方法与编程思路，本节主要介绍 ListAdapter+HashMap+ArrayList 的使用和 SimpleCursorAdapter 的使用的内容。

6.2.1 ListAdapter+HashMap+ArrayList的使用

在对 ListView 进行更为复杂的操作时，常常使用 ListAdapter+HashMap+ArrayList 组装其数据源，具体如何实现呢？一般是将数据保存在 HashMap 中，将 HashMap 添加到 ArrayList 数组对象中，将 ArrayList 对象作为 ListAdapter 数据适配器的参数传入，最后绑定到 ListView 控件上。

ListAdapter 是绑定 Data 和 Listview 的适配器。在 3.2.1 章节数据适配器（Adapter）曾阐述过，它是接口，需要使用它的子类（ArrayAdapter，SimpleAdapter，CursorAdapter）配合 HashMap、ArrayList 一起使用。

HashMap 类实现了 Map 接口，它主要功能提供一种键（key）值（value）保存数据的方式。在数组中通过数组下标来对其内容进行索引，而在 Map 中是通过对象来对对象进行索引，用来索引的对象叫做 key，其对应的对象叫做 value。它的映射关系就像表中的字段与表中的值一样，可以把表的查询结果完全转换成 HashMap 散列表。

ArrayList 对象是包含单一数据值的项目的集合，通过 Add()方法向 ArrayList 添加项目。它的默认初始容量为 0，但随着元素添加到 ArrayList 中，容量会根据需要通过重新分配自动增加。

下面通过一个示例，讲述 ListAdapter+HashMap+ArrayList 的用法，示例代码运行结果如图 6-4 所示。

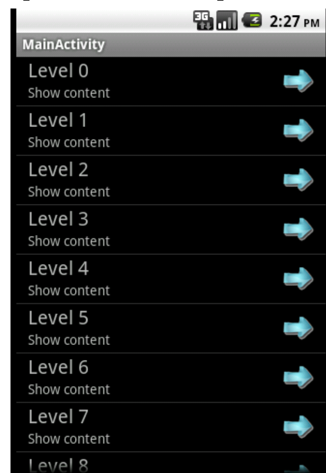


图 6-4 ListAdapter 适配器示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 6-5、代码清单 6-6 所示。

代码清单 6-5 ListAdapter+HashMap+ArrayList使用示例（第 6 章\Demo_06_03）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/LinearLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android">
```



```
        <ListView
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/ListView01"
                />
</LinearLayout>
```

代码清单 6-6　ListAdapter+HashMap+ArrayList使用示例（第 6 章\Demo_06_03）list_ items.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
        android:id="@+id/RelativeLayout01"
        android:layout_width="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:layout_height="wrap_content"
        android:paddingBottom="4dip"
        android:paddingLeft="12dip"
        android:paddingRight="12dip">
        <ImageView
                android:paddingTop="12dip"
                android:layout_alignParentRight="true"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:id="@+id/ItemImage"
                />
        <TextView
              android:text="TextView01"
              android:layout_height="wrap_content"
              android:textSize="20dip"
              android:layout_width="fill_parent"
              android:id="@+id/ItemTitle"
              />
        <TextView
                android:text="TextView02"
                android:layout_height="wrap_content"
                android:layout_width="fill_parent"
                android:layout_below="@+id/ItemTitle"
                android:id="@+id/ItemText"
                />
</RelativeLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 6-7 所示。

代码清单 6-7　ListAdapter+HashMap+ArrayList使用示例（第 6 章\Demo_06_03）MainActivity.java

```
package com.chen.android;

import android.os.Bundle;
import android.view.View;
import java.util.ArrayList;
import java.util.HashMap;
```




```
import android.app.Activity;
import android.view.ContextMenu;
import android.view.MenuItem;
import android.view.ContextMenu.ContextMenuInfo;
import android.view.View.OnCreateContextMenuListener;
import android.widget.AdapterView;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleAdapter;
import android.widget.AdapterView.OnItemClickListener;

/**
 * Demo_06_03 ArrayList +ListAdapter 数据源的构造
 *
 * @author
 */
public class MainActivity extends Activity {
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		// 绑定 Layout 里面的 ListView
		ListView list = (ListView) findViewById(R.id.ListView01);

		/* 初始化 ArrayList 数据源对象 */
		ArrayList<HashMap<String, Object>> listItem =
							new ArrayList<HashMap<String, Object>>();
		// 生成动态数组，加入数据源对象中
		for (int i = 0; i < 10; i++) {
			HashMap<String, Object> map = new HashMap<String, Object>();
			map.put("ItemImage", R.drawable.checked);// 图像资源的 ID
			map.put("ItemTitle", "Level " + i);
			map.put("ItemText", "Show content");
			listItem.add(map);
		}
		// 生成适配器的 Item 和动态数组对应的元素
		ListAdapter listItemAdapter = new SimpleAdapter(this, listItem,// 数据源
				R.layout.list_items,// ListItem 的 XML 实现
				// 动态数组与 ImageItem 对应的子项
				new String[] { "ItemImage", "ItemTitle", "ItemText" },
				// ImageItem 的 XML 文件里面的一个 ImageView,两个 TextView ID
				new int[] { R.id.ItemImage, R.id.ItemTitle, R.id.ItemText });

		// 添加并且显示
		list.setAdapter(listItemAdapter);

		// 添加单击
		list.setOnItemClickListener(new OnItemClickListener() {

			@Override
			public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
```



```
                    long arg3) {
                setTitle("单击第" + arg2 + "个项目");
            }
        });

        // 添加长按单击
        list.setOnCreateContextMenuListener(new OnCreateContextMenuListener() {

            @Override
            public void onCreateContextMenu(ContextMenu menu, View v,
                    ContextMenuInfo menuInfo) {
                menu.setHeaderTitle("长按菜单-ContextMenu");
                menu.add(0, 0, 0, "弹出长按菜单 0");
                menu.add(0, 1, 0, "弹出长按菜单 1");
            }
        });
    }

    // 长按菜单响应函数
    @Override
    public boolean onContextItemSelected(MenuItem item) {
        // 在标题栏显示选择了第几项
        setTitle("单击了长按菜单里面的第" + item.getItemId() + "个项目");
        return super.onContextItemSelected(item);
    }
}
```

6.2.2　SimpleCursorAdapter的使用

SimpleCursorAdapter 允许绑定一个游标的列到 ListView 控件上，并使用自定义的 layout 显示每个项目。可以使用 SimpleCursorAdapter 作为中间桥梁，将数据库中查询出来的数据直接显示到 ListView 中。其构造函数结构如下：

```
        public SimpleCursorAdapter(Context context， int layout， Cursor c， String[] from， int[] to)
{   super(context， layout， c);    mTo = to;    mOriginalFrom = from;    findColumns(from);
        }
```

具体参数含义如表 6-3 所示。

表 6-3　SimpleCursorAdapter()方法参数功能说明

SimpleCursorAdapter 的参数	功 能 说 明
Context　context	这个与 SimpleListItemFactory 相关的 ListView 所处运行上下文（context）也就是这个 ListView 所在的 Activity
int　layout	显示 ListView item 的布局文件。这个 layout 文件中至少要包含在"to"参数中命名的 views
Cursor　c	数据库的光标(Cursor)。如果 cursor 无效，则该参数可以为 null
String[]　from；	指定用于显示"from"参数指定的数据列表的 views，这些 views 必须都是 TextViews。"from"参数的前 *N* 个值（values）和"to"参数的前 *N* 个 views 是一一对应的关系。如果 cursor 无效，则该参数可为 null
int[]　to	指定用于显示 "to" 参数指定的数据列表的 views，这些 views 必须都是 TextViews

如何使用 SimpleCurosrAdapter 呢？它主要是将 Cursor 中的 columns 与在 XML 文件中定义的 TextViews 或 ImageViews 进行匹配的简易适配器（Adapter）。你可以指定选择 Cursor 中的哪些 columns、用哪些 views 来显示这些 columns 以及指定定义这些 views 的 XML 文件。也就是说，SimpleCursorAdapter 允许绑定一个 Cursor 的 columns 到 ListView 上，并使用自定义的 layout 显示 ListView 中的每个项目。下面通过一个示例，讲述 SimpleCurosrAdapter 的用法，示例代码运行结果如图 6-5 所示。



图 6-5 SimpleCursorAdapter 适配器示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 6-8 所示。

代码清单 6-8 SimpleCursorAdapter适配器使用示例（第 6 章\Demo_06_04）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout
          xmlns:android="http://schemas.android.com/apk/res/android"
           android:orientation="vertical"
           android:layout_width="fill_parent"
           android:layout_height="fill_parent">

          <ListView android:id="@id/android:list"
                    android:layout_width="fill_parent"
                    android:layout_height="fill_parent"
                    android:layout_weight="1"
           />

          <TextView android:id="@id/android:empty"
                    android:layout_width="fill_parent"
                    android:layout_height="wrap_content"
                    android:text="No data"
                    android:textColor="#ff0000"
           />
 </LinearLayout>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现这一效果，如代码清单 6-9 所示。

代码清单 6-9 SimpleCursorAdapter适配器使用示例（第 6 章\Demo_06_04）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.database.Cursor;
import android.graphics.Color;
import android.os.Bundle;
```



```
import android.provider.Contacts.Phones;
import android.view.View;
import android.widget.AdapterView;
import android.widget.LinearLayout;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;

/**
 * SimpleCursorAdapter 的使用
 *
 */
public class MainActivity extends Activity {
	LinearLayout m_LinearLayout;
	ListView m_ListView;

	/** 当 activity 首次创建时响应调用 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);

		/* 创建 LinearLayout 布局对象 */
		m_LinearLayout = new LinearLayout(this);
		/* 设置布局 LinearLayout 的属性 */
		m_LinearLayout.setOrientation(LinearLayout.VERTICAL);
		m_LinearLayout.setBackgroundColor(android.graphics.Color.BLACK);

		/* 创建 ListView 对象 */
		m_ListView = new ListView(this);
		LinearLayout.LayoutParams param = new LinearLayout.LayoutParams(
				LinearLayout.LayoutParams.FILL_PARENT,
				LinearLayout.LayoutParams.WRAP_CONTENT);
		m_ListView.setBackgroundColor(Color.BLACK);

		/* 添加 m_ListView 到 m_LinearLayout 布局 */
		m_LinearLayout.addView(m_ListView, param);

		/* 设置显示 m_LinearLayout 布局 */
		setContentView(m_LinearLayout);

		// 获取数据库 Phones 的 Cursor
		Cursor c = getContentResolver().query(Phones.CONTENT_URI, null, null,
				null, null);
		startManagingCursor(c);

		// ListAdapter 是 ListView 和后台数据的桥梁
		ListAdapter adapter = new SimpleCursorAdapter(this,
		// 定义 List 中每一行的显示模板
		// 表示每一行包含两个数据项
```




```
                    android.R.layout.simple_list_item_2,
                    // 数据库的 Cursor 对象
                    c,
                    // 从数据库的 NAME 和 NUMBER 两列中取数据
                    new String[] { Phones.NAME, Phones.NUMBER },
                    // 与 NAME 和 NUMBER 对应的 Views
                    new int[] { android.R.id.text1, android.R.id.text2 });

        /* 将 adapter 添加到 m_ListView 中 */
        m_ListView.setAdapter(adapter);

        /* 为 m_ListView 视图添加 setOnItemSelectedListener 监听 */
        m_ListView.setOnItemSelectedListener(ItemSelected);
        m_ListView.setOnItemClickListener(OnItemClick);

    }

    /**
     * 实现对 ItemSelected 监听
     */
    ListView.OnItemSelectedListener ItemSelected
                            = new ListView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> arg0, View view,
                int position, long id) {
            // TODO Auto-generated method stub
            DisplayToast("滚动到第" + Long.toString(arg0.getSelectedItemId()) + "项");
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // TODO Auto-generated method stub
        }

    };

    /**
     * 实现对 ItemClick 监听
     */
    ListView.OnItemClickListener OnItemClick = new ListView.OnItemClickListener() {

        @Override
        public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                long arg3) {
            // 对选中的项进行处理
            DisplayToast("选中了第" + Integer.toString(arg2 + 1) + "项");
        }
    };

    /* 显示 Toast */
    public void DisplayToast(String str) {
        Toast.makeText(this, str, Toast.LENGTH_SHORT).show();
```



```
        }
    }
```

6.3　选择功能实现

选择功能要实现的样式效果如前面的章节所示，在布局中使用到了 ListView 控件，需要注意的是要学会如何为其添加子项样式及其数据源，相信通过本章知识基础和重点剖析部分的讲解及大量示例的练习，大家深有体会。

6.3.1　选择界面View实现

根据第 2 章手机订票系统实现效果图（图 2-6），城市选择界面采用 LinearLayout 线性布局，易于实现。详见代码清单 6-10、代码清单 6-11。

代码清单 6-10　第 6 章城市选择界面示例（第 6 章\GanaSkyForSelectCity）citylistmain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/LinearLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    >

    <AutoCompleteTextView
        android:id="@+id/AutoCompleteTextView_City"
        android:layout_width="292dip"
        android:layout_height="wrap_content"
        android:padding="15dip"
        android:textSize="15sp"
        android:hint="输入城市首字母进行搜索"
    >
    </AutoCompleteTextView>
    <TextView
        android:id="@+id/widget31"
        android:layout_width="294dip"
        android:layout_height="24dip"
        android:text="热门城市"
        android:layout_gravity="center_horizontal"
    >
    </TextView>

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ListView01"/>
</LinearLayout>
```

代码清单 6-11　第 6 章注册界面示例（第 6 章\GanaSkyForSelectCity）citylist_items.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    android:id="@+id/RelativeLayout01"
     android:layout_width="fill_parent"
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_height="wrap_content"
     android:paddingBottom="4dip"
     android:paddingLeft="12dip"
     android:paddingRight="12dip" >

     <TextView
          android:text="TextView02"
          android:layout_height="wrap_content"
           android:layout_width="fill_parent"
           android:paddingLeft="100dip"
           android:id="@+id/ItemText"
           android:textSize="15sp"
           android:layout_centerVertical="true"/>

</RelativeLayout>
```

6.3.2　选择功能Model类实现

选择功能 Model 类实现的方法，请参考代码清单 6-12。

代码清单 6-12　第 6 章城市选择界面示例（第 6 章\GanaSkyForSelectCity）City.java

```
package com.ganasky.model;
/**
 * 城市类
 * @author
 *
 */
public class City {

     private    String strCode;
     private String strName;
     private String strPinYin;
     private String strShort;

     public String getCode() {
          return strCode;
     }
     public void setCode(String strCode) {
          this.strCode = strCode;
     }
     public String getName() {
          return strName;
     }
```



```
        public void setName(String strName) {
                this.strName = strName;
        }
        public String getPinYin() {
                return strPinYin;
        }
        public void setPinYin(String strPinYin) {
                this.strPinYin = strPinYin;
        }
        public String getShort() {
                return strShort;
        }
        public void setShort(String strShort) {
                this.strShort = strShort;
        }

}
```

6.3.3　选择功能Control实现

选择功能的 Control 实现，就是城市选择功能的 Activity 代码实现，如代码清单 6-13 所示。在实现过程中需要注意的有以下几点：

1）如何接收与解析 ASP 后台传来的数据，并保存到 City 类里，再作为 ListView 类的子项显示出来。这一部分是实现的难点，在本章中的难点部分（ListAdapter+ HashMap+ ArrayList 的使用）已详细讲解。

2）如何输入首字母自动匹配城市，这里使用 AutoCompleteTextView 实现自动匹配城市拼音，选择出发和到达城市。

3）如何读取 ListView 的子项内容，并作为参数返回到 TicketMainApp.java 界面，也是实现的重点。

代码清单 6-13　第 6 章选择功能Control实现（第 6 章\GanaSkyForSelectCity）CityListView.java（详见本书源代码）

6.3.4　选择功能后台ASP实现

选择功能后台 ASP 实现，见代码清单 6-14。

代码清单 6-14　第 6 章选择功能ASP实现（第 6 章\GanaSkyForSelectCity）CityQueryPage. aspx.cs（详见本书源代码）

城市选择功能执行 ASP 后台代码后返回的 XML 格式数据，如代码清单 6-15 所示。这里只列出部分数据，要强调的是数据组装的格式，在代码中有全部数据。

代码清单 6-15　注册功能执行ASP后台代码后返回的XML格式数据（第 6 章\GanaSkyForSelectCity\assets\data）city.xml（部分数据）

```
<?xml version="1.0" encoding="utf-8"?>
<CityInfo>
  <City>
```




```
        <Code>AAT</Code>
        <Name>阿勒泰</Name>
        <PinYin>ALTAY</PinYin>
        <Short>ALT</Short>
    </City>
    <City>
        <Code>ACX</Code>
        <Name>兴义</Name>
        <PinYin>XINGYI</PinYin>
        <Short>XY</Short>
    </City>
    <City>
        <Code>AKA</Code>
        <Name>安康</Name>
        <PinYin>ANKANG</PinYin>
        <Short>AK</Short>
    </City>
    <City>
        <Code>AKL</Code>
        <Name>奥克兰</Name>
        <PinYin>AUCKLAND</PinYin>
        <Short>AKL</Short>
    </City>
</CityInfo>
```

第 7 章　查询功能实现

如第 2 章所显示的项目实现效果图 2-7 所示，本章涉及的基础知识有：图片视图（ImageView）、图片按钮（ImageButton）、下拉列表（Spinner）、日期和时间控件（DatePicker、TimePicker）、菜单（Menu）、对话框（Dialog）及进度条（ProgressBar）控件的使用等，知识点相对较多。

本章知识的难点是：查询要涉及 Dialog 和线程的使用，如何使用线程（Thread）handler 和 Runnable，将是本章难点部分要讲述的。

7.1　基础控件讲解

本节涉及图片视图（ImageView）、图片按钮（ImageButton）、下拉列表（Spinner）、日期和时间控件（DatePicker、TimePicker）、菜单（Menu）、对话框（Dialog）及进度条（ProgressBar）等控件，这些控件可以使用在查询功能中，也可以使用在其他功能和其他项目中。

7.1.1　图片视图

图片视图（ImageView）的使用无处不在，在网站中插入图片可以为程序的界面增色不少，在之前的示例中也曾多次使用过 ImageView 控件，在这里详细讲解其主要属性、方法及其用法。一般建议在程序中加入图片资源，而不是放在 SD 卡中以流的方式去读取图片资源，这样做的好处在于嵌入的资源比较安全，不容易被篡改。从程序中读取图片文件，一般将图片文件保存在项目 res\drawable 开头的 3 个文件夹下（如图 7-1），他们分别代表了高、中、低分辨率的图片。Android 在读取图片时将会自动优化，选用适合特定手机分辨率的一个图片显示，比如高分辨率可以存放 128*128 的图片，低分辨率可以存放 32*32 的图片。也可以不区分不同分辨率的图片，统一放在 res\drawable 文件夹下。

ImageView 控件的一个重要属性就是设置图片源（android:src），如果把图片放在 res\drawable 开头的文件夹下，在 java 程序中可以直接引用图片的名字，从而加载该图片。例如：android:src="@drawable/picture"，就是引用 picture.jpg 这张图片。

ImageView 图片视图的另一个重要属性是 scaleType（android:scaleType），它是控制图片如何放大或者移动来匹配 ImageView 的尺寸大小，在之前的示例中也曾涉及，在这里详细讲解 ImageView.ScaleType（在 java Activity 中的定义方法）和 android:scaleType（在 xml 属性中的定义方法）值的具体含义区别，见表 7-1。

表 7-1　ImageView.ScaleType 和 android:scaleType 值类型区别

ImageView.ScaleType/ android:scaleType 值类型	功 能 说 明
CENTER /center	按图片原来的 size（尺寸大小）居中显示，当图片长/宽超过 View（容器）的长/宽时，则截取图片的居中部分显示
CENTER_CROP / centerCrop	按比例扩大图片的 size 居中显示，使得图片长（宽）等于或大于 View 的长（宽）
CENTER_INSIDE/ centerInside	将图片的内容完整居中显示，通过按比例缩小或原来的 size 使得图片长/宽等于或小于 View 的长/宽
FIT_CENTER / fitCenter	把图片按比例扩大/缩小到 View 的宽度，居中显示
FIT_END / fitEnd	把图片按比例扩大/缩小到 View 的宽度，显示在 View 的下部分位置
FIT_START / fitStart	把图片按比例扩大/缩小到 View 的宽度，显示在 View 的上部分位置
FIT_XY / fitXY	把图片不按比例扩大/缩小到 View 的大小显示
MATRIX / matrix	把图片用矩阵来绘制

下面通过一个示例讲述图片视图 ImageView 的用法，代码运行结果如图 7-2 所示。

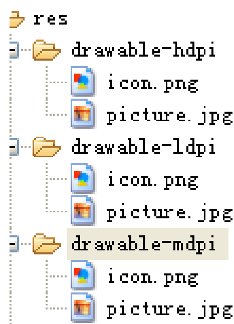


图 7-1　程序中 3 个图片文件夹



图 7-2　ImageView 图片视图示例运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-1 所示。

代码清单 7-1　ImageView显示图片示例（第 7 章\Demo_Demo_07_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

    <TextView
    android:id="@+id/TextView01"
    android:text="显示图片："
    android:layout_width="wrap_content"
    android:layout_height="wrap_content"
    />
```



```
        <ImageView
            android:id="@+id/ImageView01"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:layout_below="@id/TextView01"
            android:scaleType="centerCrop"
            />

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现显示图片这一功能，如代码清单 7-2 所示。

代码清单 7-2　ImageView显示图片示例（第 7 章\Demo_07_01）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.widget.ImageView;
import android.widget.TextView;

/**
 * Demo_07_01　ImageView 的使用
 * @author
 *
 */
public class MainActivity extends Activity {
    //  声明 ImageView 对象
    ImageView imageview;
    //声明 Textview 对象
    TextView textview;
    // ImageView 的 alpha 值
    int image_alpha = 255;
  //线程
    Handler mHandler = new Handler();
    //  控件线程
    boolean isrung = false;

    /**当 activity 首次创建时响应调用*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        isrung = true;
```

```
            // 获得 ImageView 的对象
            imageview = (ImageView) this.findViewById(R.id.ImageView01);
            textview = (TextView) this.findViewById(R.id.TextView01);

            /*
             *   设置 imageview 的图片资源
             *   同样可以在 xml 布局中这样写： android:src="@drawable/bg"
             */
            imageview.setImageResource(R.drawable.bg);

            // 设置 imageview 的 Alpha 值
            imageview.setAlpha(image_alpha);

        }
    }
```

7.1.2 图片按钮

图片按钮（ImageButton）与 Button 用法类似，具有按钮的功能。而添加该控件的图片源方法与 ImageView 控件类似，都有 android:src 属性，在 Android 中的运用也非常灵活，既可以在 Activity java 代码中实现，也可以在 xml 布局中实现，具体实现方式如下：

（1）在 Activity java 代码中加载图片源

```
m_ImageButton.setImageDrawable(getResources().getDrawable(R.drawable.my_button));
```

（2）在 xml 布局中加载图片源

```
Android:src= "../../@drawable/ic_media_play"或 android:src="@drawable/picture"
```

下面通过一个示例讲述 ImageButton 图片按钮的用法，该示例的主要功能是实现两种方式加载图片数据源以及如何读取 Android 内置的图片，代码运行结果如图 7-3 所示。

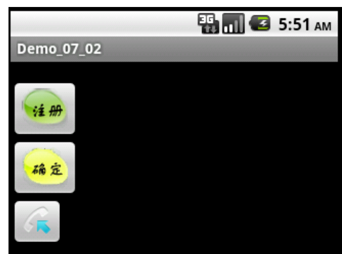


图 7-3 ImageButton 图片按钮示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-3 所示。

代码清单 7-3 ImageButton图片按钮使用示例（第 7 章\Demo_07_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
```



```
        <TextView
            android:id="@+id/TextView01"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            />
          <ImageButton
          android:id="@+id/ImageButton01"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:src="@drawable/button1"
          >
          </ImageButton>
          <ImageButton
          android:id="@+id/ImageButton02"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          >
          </ImageButton>
          <ImageButton
          android:id="@+id/ImageButton03"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          >
          </ImageButton>

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 7-4 所示。

代码清单 7-4　ImageButton图片按钮使用示例（第 7 章\Demo_07_02）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.ImageButton;
import android.widget.TextView;
import android.widget.Toast;
/**
 * Demo_07_02 图片按钮 ImageButton 实现
 * @author
 *
 */
public class MainActivity extends Activity {
    //     声明 1 个 TextView 对象
    TextView m_TextView;
    // 声明 3 个 ImageButton 对象
    ImageButton m_ImageButton1;
    ImageButton m_ImageButton2;
```




```
ImageButton m_ImageButton3;


/** 当 activity 首次创建时响应调用 */
@Override
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        m_TextView = (TextView) findViewById(R.id.TextView01);
        // 分别取得 4 个 ImageButton 对象
        m_ImageButton1 = (ImageButton) findViewById(R.id.ImageButton01);
        m_ImageButton2 = (ImageButton) findViewById(R.id.ImageButton02);
        m_ImageButton3 = (ImageButton) findViewById(R.id.ImageButton03);

        // 分别设置所使用的图标
        // m_ImageButton1 是在 xml 布局中设置的，这里就暂时不设置了
        m_ImageButton2.setImageDrawable(getResources().getDrawable(
                R.drawable.confirm));
        m_ImageButton3.setImageDrawable(getResources().getDrawable(
                android.R.drawable.sym_call_incoming));

        // 以下分别为每个按钮设置事件监听 setOnClickListener
        m_ImageButton1.setOnClickListener(OnClick1);
        m_ImageButton2.setOnClickListener(OnClick2);
        m_ImageButton3.setOnClickListener(OnClick3);
}

ImageButton.OnClickListener OnClick1 = new ImageButton.OnClickListener() {

        @Override
        public void onClick(View v) {
                // 显示提示信息
                DisplayToast("这是 Button1");
        }
};

ImageButton.OnClickListener OnClick2 = new ImageButton.OnClickListener() {

        @Override
        public void onClick(View v) {
                // 显示提示信息
                DisplayToast("这是 Button2");
        }
};
ImageButton.OnClickListener OnClick3 = new ImageButton.OnClickListener() {

        @Override
        public void onClick(View v) {
                // 显示提示信息
```



```
                    DisplayToast("这是使用系统图标");
                }
            };

            /*  显示 Toast */
            public void DisplayToast(String str) {
                Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
                //  设置 toast 显示的位置
                toast.setGravity(Gravity.TOP, 0, 220);
                //  显示该 toast
                toast.show();
            }
        }
```

7.1.3　下拉列表

下拉列表（Spinner）是一种下拉列表控件，单击控件弹出一个对话框，显示如同 ListView 的列表选项，它位于 android.widget 包下，每次只显示用户选中的元素。当用户单击 Spinner 选择箭头时，会弹出列表供用户选择，而选择列表中的元素同样来自适配器（ArrayAdapter）可将菜单项（数据源）放在适配器中，调用 setAdapter()方法加载数据源。Spinner 控件的数据源还可以使用 xml 文件，可在 values 文件夹下新建一个 arryas.xml 文件，然后在 Activity.java 代码中调用此 xml 文件。

下面通过一个示例讲述下拉列表 Spinner 的用法，该示例的主要功能是动态添加、删除 Spinner 数据源，单击 Spinner 控件显示子项，代码运行结果如图 7-4 至图 7-6 所示。

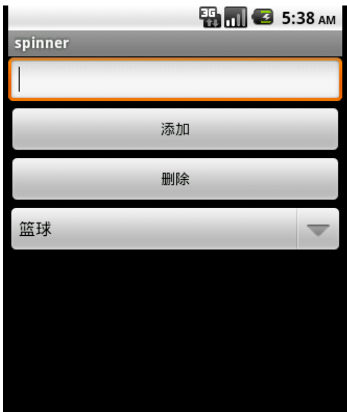


图 7-4　下拉列表 Spinner 示例代码运行结果（1）——初始化



图 7-5　下拉列表 Spinner 示例代码运行结果（2）——Spinner 弹出样式



图 7-6　下拉列表 Spinner 示例代码运行结果（3）——为 Spinner 增加子项

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-5、代码清单 7-6 所示。

代码清单 7-5　下拉列表Spinner使用示例（第 7 章\Demo_07_03）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
```

```
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
         <EditText android:id="@+id/et"
               android:layout_width="fill_parent"
             android:layout_height="wrap_content"
         />
         <Button android:id="@+id/add"
               android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="添加"
         />
         <Button android:id="@+id/remove"
               android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:text="删除"
         />
         <Spinner android:id="@+id/sp"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
         />
</LinearLayout>
```

代码清单 7-6　下拉列表Spinner使用示例（第 7 章\Demo_07_03）strings.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
        <string name="hello">Hello World, main!</string>
        <string name="app_name">spinner</string>
        <string-array name="action">
         <item>篮球</item>
         <item>足球</item>
         <item>排球</item>
        </string-array>
</resources>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 7-7 所示。

代码清单 7-7　下拉列表Spinner使用示例（第 7 章\Demo_07_03）MainActivity.java

```
package com.chen.android;

import java.util.ArrayList;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.ArrayAdapter;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Spinner;
```



```
/**
 * Demo_07_03 Spinner 的使用方法
 * @author
 *
 */
public class MainActivity extends Activity {

      // 声明 EditText 对象
      private EditText et;
      // 声明 Button 对象
      private Button add,remove;
      // 声明 Spinner 对象
      private Spinner sp;
      // 声明 ArrayList 对象
      ArrayList<String> arrayList=new ArrayList<String>();
      ArrayAdapter<String> adapter;
     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);

          /*获取 xml 布局中的控件*/
          et=(EditText)findViewById(R.id.et);
          add=(Button)findViewById(R.id.add);
          remove=(Button)findViewById(R.id.remove);
          sp=(Spinner)findViewById(R.id.sp);

          // 从 res/values 中获取所定义的数组
          String[]  arrList=getResources().getStringArray(R.array.action);
          //获取 xml 中定义的数组
          for(int i=0;i<arrList.length;i++){
            arrayList.add(arrList[i]);
          }
          //把数组加载到 ArrayList 中
       adapter=new ArrayAdapter<String>(this,android.R.layout.simple_spinner_item,arrayList);
        //设置下拉菜单的风格
       adapter.setDropDownViewResource(android.R.layout.simple_spinner_dropdown_item);
        //绑定适配器
          sp.setAdapter(adapter);
        //设置列表框 spinner 标题
          sp.setPrompt("您喜欢的运动：");

          //添加、删除按钮监听器
          add.setOnClickListener(addSpinnerValue);
          remove.setOnClickListener(deleteSpinnerValue);
     }

     /**
      * 实现添加子项监听方法
      */
     Button.OnClickListener addSpinnerValue= new Button.OnClickListener(){

            @Override
```




```
            public void onClick(View v) {
                /*添加输入的项，调用 add 方法后自动调用 notifyDataSetChanged()
                *如果需要指定位置，使用 insert(String s, int index)方法
                */
                adapter.add(et.getText().toString());
                //在标题输出框中添加后 list 的大小
                setTitle(String.valueOf(arrayList.size()));
            }
        };

        /**
         * 实现删除子项监听方法
         */
        Button.OnClickListener deleteSpinnerValue= new Button.OnClickListener(){

            @Override
            public void onClick(View v) {
                //删除当前选中项,remove 后自动调用 notifyDataSetChanged()
                adapter.remove(sp.getSelectedItem().toString());
                //在标题输出框中添加后 list 的大小
                setTitle(String.valueOf(arrayList.size()));
            }

        };
    }
```

7.1.4 日期和时间控件

日期和时间控件（DatePicker、TimePicker）可以让用户选择日期，免去判断输入日期的合法性。它继承自 FrameLayout 类，日期选择控件的主要功能是向用户提供包含年、月、日的日期数据并允许用户对其进行修改。如果要捕获用户修改日期后日期控件中的数据事件，需要为 DatePicker 添加 OnDateChangedListener 事件监听器，实现 onDateChanged()方法。

TimePicker 也继承自 FrameLayout 类，它是允许用户选择时间。它主要是向用户显示一天中的时间（可以为 24h，也可以为 AM/PM 制）并允许用户进行选择。如果要捕获用户修改时间后时间控件中的数据事件，也需要为 TimePicker 添加 OnTimeChangedListener 事件监听器，实现 onTimeChanged()方法。

DatePicker、TimePicker 控件可以配合 Calendar（日历）类的使用，它是为设定年度日期对象和一个整数字段之间转换的抽象基类，如：年、月、日、时、分等。Calendar 常见的方法如表 7-2 所示。

表 7-2　Calendar 类读取日期和时间

实例化 Calendar	final Calendar calendar=Calendar.getInstance();
调用 calendar 相应方法	mYear=calendar.get(Calendar.YEAR);——获取年份
	mMonth=calendar.get(Calendar.MONTH);——获取月份
	mDay=calendar.get(Calendar.DAY_OF_MONTH);——获取日（哪一天）
	mHour=calendar.get(Calendar.HOUR_OF_DAY);——获取时
	mMinute=calendar.get(Calendar.MINUTE);——获取分



下面通过一个示例讲述日期和时间控件的用法，示例的主要功能是实现两种方式使用日期和时间控件，代码运行结果如图 7-7 至图 7-9 所示。



图 7-7 日期和时间控件示例代码运行结果（1）——初始化

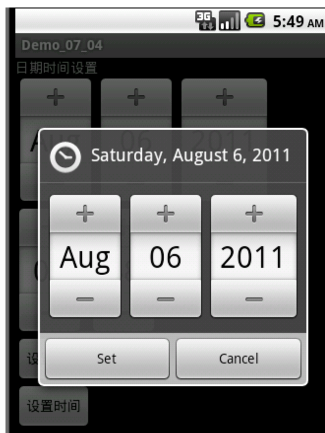


图 7-8 日期和时间控件示例代码运行结果（2）——设置日期

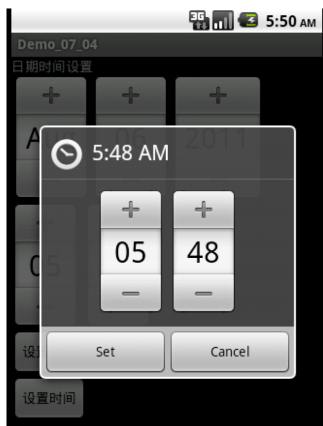


图 7-9 日期和时间控件示例代码运行结果（3）——设置时间

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-8 所示。

代码清单 7-8 日期和时间控件使用示例（第 7 章\Demo_07_04）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<TextView
      android:id="@+id/TextView01"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
    />
    <DatePicker
      android:id="@+id/DatePicker01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
    >
    </DatePicker>
    <TimePicker
      android:id="@+id/TimePicker01"
      android:layout_width="wrap_content"
      android:layout_height="wrap_content"
    >
    </TimePicker>
    <Button
      android:id="@+id/button1"
```

```
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="设置日期"
      >
      </Button>
      <Button
        android:id="@+id/button2"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:text="设置时间"
      >
      </Button>
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 7-9 所示。

代码清单 7-9　日期和时间控件使用示例（第 7 章\Demo_07_04）MainActivity.java

```
package com.chen.android;

import java.util.Calendar;
import android.app.Activity;
import android.app.DatePickerDialog;
import android.app.TimePickerDialog;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.DatePicker;
import android.widget.TextView;
import android.widget.TimePicker;

/**
 * Demo_07_04 日期和时间控件的用法
 *
 * @author
 *
 */
public class MainActivity extends Activity {
        TextView m_TextView;
        // 声明 DatePicker 对象
        DatePicker m_DatePicker;
        // 声明 TimePicker 对象
        TimePicker m_TimePicker;
        //声明 Button 对象
        Button m_dpButton;
        Button m_tpButton;
        // java 中的日历类
        Calendar c;

        /** 主程序*/
        @Override
```



```
public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
    /*实例化一个日历类*/
        c = Calendar.getInstance();
    /*获取 xml 布局中的控件*/
        m_TextView = (TextView) findViewById(R.id.TextView01);
        m_dpButton = (Button) findViewById(R.id.button1);
        m_tpButton = (Button) findViewById(R.id.button2);

        // 获取 DatePicker 对象
        m_DatePicker = (DatePicker) findViewById(R.id.DatePicker01);
        // 将日历初始化为当前系统时间，并设置其事件监听
        m_DatePicker.init(c.get(Calendar.YEAR), c.get(Calendar.MONTH),
                c.get(Calendar.DAY_OF_MONTH),initDate);

        // 获取 TimePicker 对象
        m_TimePicker = (TimePicker) findViewById(R.id.TimePicker01);
        // 设置为 24h 制显示
        m_TimePicker.setIs24HourView(true);

        // 监听时间改变
        m_TimePicker.setOnTimeChangedListener(timeChange);
        // 监听按钮弹出日期选择对话框
        m_dpButton.setOnClickListener(dpbuttonClick);
        // 监听按钮弹出时间选择对话框
        m_tpButton.setOnClickListener(tpbuttonClick);
}

/**
  * 实现监听
  */
DatePicker.OnDateChangedListener initDate
                                    = new DatePicker.OnDateChangedListener() {

        @Override
        public void onDateChanged(DatePicker view, int year, int monthOfYear,
                int dayOfMonth) {
            // 当日期更改时，在这里处理
            // c.set(year, monthOfYear, dayOfMonth)

        }
};

TimePicker.OnTimeChangedListener timeChange
                                    = new TimePicker.OnTimeChangedListener() {

        @Override
        public void onTimeChanged(TimePicker view, int hourOfDay, int minute) {
            // 时间改变时处理
            // c.set(hourOfDay, minute)
```

```
                }

            };

            Button.OnClickListener   dpbuttonClick
                            = new Button.OnClickListener() {
                @Override
                public void onClick(View v) {
                    // 弹出一个时间对话框
                    new DatePickerDialog(MainActivity.this,
                            new DatePickerDialog.OnDateSetListener() {
                                public void onDateSet(DatePicker view, int year,
                                        int monthOfYear, int dayOfMonth) {
                                    // 设置日期
                                    // c.set(year, monthOfYear, dayOfMonth)
                                }
                            }, c.get(Calendar.YEAR), c.get(Calendar.MONTH),
                            c.get(Calendar.DAY_OF_MONTH)).show();
                }

            };

            Button.OnClickListener   tpbuttonClick = new Button.OnClickListener() {

                @Override
                public void onClick(View v) {
                    // 弹出一个时间对话框
                    new TimePickerDialog(MainActivity.this,
                            new TimePickerDialog.OnTimeSetListener() {
                                public void onTimeSet(TimePicker view, int hourOfDay,
                                        int minute) {
                                    // 设置时间
                                    // c.set(hourOfDay, minute);
                                }
                            }, c.get(Calendar.HOUR_OF_DAY), c.get(Calendar.MINUTE),
                            true).show();
                }

            };
        }
```

7.1.5 菜单

在 Android 中每一个视图组件的创建方式几乎都有两种方式：

1）在 xml 布局文件中创建，通过此方式创建的组件可以通过 findViewById()方法获取 android:id 的方式设置其属性。

2）在 Activity java 代码中通过写 java 代码创建，在 Activity java 代码中可以通过 set 方式调用相对应的组件属性。

Menu 也不例外，既可以通过 xml 布局添加菜单，也可以在 Activity java 代码中添加菜单。

通常，菜单（Menu）控件不需要添加在 xml 布局文件中，而需要在 Activity java 代码



中重写 onCreateOptionsMenu(Menu menu)方法，当我们在模拟器或者在手机上单击 Menu 按钮时，Android 系统会自动调用这个函数，生成相应的菜单选项。在这里，还需要重写 onOptionsItemSelected(MenuItem item)方法，当用户选择某个菜单项时会调用这个函数，从而实现对应的操作。

在 Android 中 Menu 被分为三种类型：选项菜单（Option Menu）、上下文菜单（Context Menu）和子菜单（Sub Menu）使用最多的是选项菜单（Option Menu）。Android 手机上还有个 Menu 按键，当 Menu 按下的时候，每个 Activity 都可以选择处理这一请求，在屏幕底部弹出一个菜单，这个菜单我们就叫它选项菜单 OptionsMenu，一般情况下，选项菜单最多显示 2 排，每排 3 个菜单项，这些菜单项有文字有图标，也被称做 Icon Menus，如果多于 6 项，从第 6 项开始会被隐藏，第 6 项会出现一个 More 里，单击 More 才出现第 6 项以及以后的菜单项，这些菜单项也被称做 Expanded Menus。

下面通过一个示例讲述菜单 Menu 的用法，该示例提供了两种方式的 Menu 创建方法，代码运行结果如图 7-10 至图 7-12 所示。

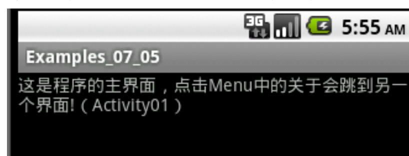


图 7-10　菜单 Menu 示例代码运行结果（1）——初始化



图 7-11　菜单 Menu 示例代码运行结果（2）——单击 Menu 弹出菜单项

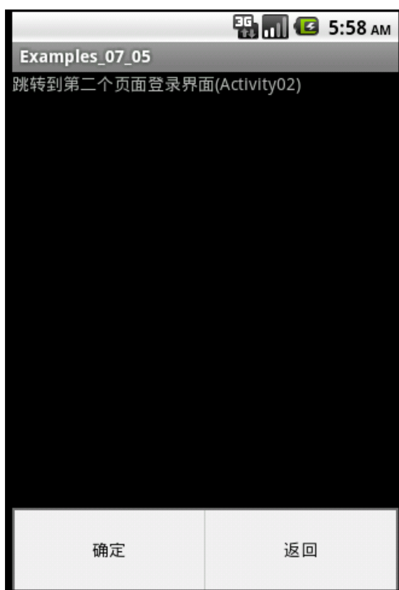


图 7-12　菜单 Menu 示例代码运行结果（3）——单击登录弹出第二屏

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-10 至代码清单 7-13 所示。

代码清单 7-10　菜单Menu使用示例（第 7 章\Demo_07_05）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent"
        >
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/main"
        />
    </LinearLayout>
```

代码清单 7-11　菜单Menu使用示例（第 7 章\Demo_07_05）main2.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <LinearLayout
       xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
         <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/login"
        >
        </TextView>
    </LinearLayout>
```

代码清单 7-12　菜单Menu使用示例（第 7 章\Demo_07_05）main3.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >
         <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/register"
        >
        </TextView>
    </LinearLayout>
```

代码清单 7-13　菜单Menu使用示例（第 7 章\Demo_07_05\layout\ menu）menu.xml

```
    <menu xmlns:android="http://schemas.android.com/apk/res/android">
        <item android:id="@+id/login"
              android:title="登录" />
        <item android:id="@+id/register"
```



```
                android:title="注册" />
        <item android:id="@+id/exit"
                android:title="退出" />
</menu>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 7-14 至代码清单 7-16 所示。

代码清单 7-14　菜单Menu使用示例（第 7 章\Demo_07_05）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuInflater;
import android.view.MenuItem;
/**
 * Demo_07_05 Menu 的用法
 * @author
 *
 */
public class MainActivity extends Activity {
        /** 当 activity 首次创建时响应调用 */
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
        }

        /* 创建 menu */
        public boolean onCreateOptionsMenu(Menu menu) {
                /*用来解析定义在 menu 目录下的菜单布局文件*/
                MenuInflater inflater = getMenuInflater();
                // 获取 menu 界面（为 res/menu/menu.xml）
                inflater.inflate(R.menu.menu, menu);
                return true;
        }

     /**
     *        处理菜单事件
     */
        public boolean onOptionsItemSelected(MenuItem item) {
                // 得到当前选中的 MenuItem 的 ID,
                int item_id = item.getItemId();

                switch (item_id) {
```

```
            case R.id.login:
                /* 新建一个 Intent 对象 */
                Intent intent = new Intent();
                /* 指定 intent 要启动的类 */
                intent.setClass(MainActivity.this, Login.class);
                /* 启动一个新的 Activity */
                startActivity(intent);
                /* 关闭当前的 Activity */
                MainActivity.this.finish();
                break;
            case R.id.register:
                /* 新建一个 Intent 对象 */
                Intent intent3 = new Intent();
                /* 指定 intent 要启动的类 */
                intent3.setClass(MainActivity.this, Register.class);
                /* 启动一个新的 Activity */
                startActivity(intent3);
                /* 关闭当前的 Activity */
                MainActivity.this.finish();
                break;
            case R.id.exit:
                MainActivity.this.finish();
                break;
            }
            return true;
        }
}
```

代码清单 7-15　菜单Menu使用示例（第 7 章\Demo_07_05）login.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
/**
 * 登录页面
 * @author
 *
 */
public class Login extends Activity {
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        /* 设置显示 main2.xml 布局 */
        setContentView(R.layout.main2);
    }
```



```
        /* 创建 menu */
        public boolean onCreateOptionsMenu(Menu menu) {
                // 为 menu 添加内容
                menu.add(0, 0, 0, R.string.ok);
                menu.add(0, 1, 1, R.string.back);
                return true;
        }

        /* 处理 menu 的事件 */
        public boolean onOptionsItemSelected(MenuItem item) {
                // 得到当前选中的 MenuItem 的 ID
                int item_id = item.getItemId();

                switch (item_id) {
                case 0:
                case 1:
                        /* 新建一个 Intent 对象 */
                        Intent intent = new Intent();
                        /* 指定 intent 要启动的类 */
                        intent.setClass(Login.this, MainActivity.class);
                        /* 启动一个新的 Activity */
                        startActivity(intent);
                        /* 关闭当前的 Activity */
                        Login.this.finish();
                        break;
                }
                return true;
        }
}
```

代码清单 7-16　菜单Menu使用示例（第 7 章\Demo_07_05）Register.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.Menu;
import android.view.MenuItem;
/**
 * 注册
 * @author
 *
 */
public class Register extends Activity {
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                /* 设置显示 main2.xml 布局 */
                setContentView(R.layout.main3);
        }
```




```
        /* 创建 menu */
        public boolean onCreateOptionsMenu(Menu menu) {
                // 为 menu 添加内容
                menu.add(0, 0, 0, R.string.ok);
                menu.add(0, 1, 1, R.string.back);
                return true;
        }

        /* 处理 menu 的事件 */
        public boolean onOptionsItemSelected(MenuItem item) {
                // 得到当前选中的 MenuItem 的 ID,
                int item_id = item.getItemId();

                switch (item_id) {
                case 0:
                case 1:
                        /* 新建一个 Intent 对象 */
                        Intent intent = new Intent();
                        /* 指定 intent 要启动的类 */
                        intent.setClass(Register.this, MainActivity.class);
                        /* 启动一个新的 Activity */
                        startActivity(intent);
                        /* 关闭当前的 Activity */
                        Register.this.finish();
                        break;
                }
                return true;
        }
}
```

最后，需要在 AndroidManifest.xml 要添加访问权限，具体如代码清单 7-17 所示。

代码清单 7-17　菜单Menu使用示例（第 7 章\07_17）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest
        xmlns:android="http://schemas.android.com/apk/res/android"
        package="com.chen.android"
        android:versionCode="1"
        android:versionName="1.0"
        >
        <application android:icon="@drawable/icon" android:label="@string/app_name">
                <activity
                        android:name=".MainActivity"
                        android:label="@string/app_name"
                        >
                <intent-filter>
                <action android:name="android.intent.action.MAIN" />
        <category android:name="android.intent.category.LAUNCHER" />
                </intent-filter>
                </activity>
                <activity android:name="Login" ></activity>
                <activity android:name="Register" ></activity>
```



```
        </application>
      <uses-sdk android:minSdkVersion="8" />
   </manifest>
```

7.1.6　对话框

在讲解对话框（Dialog）使用方法之前，先讨论 Activity 如何管理对话框。Activity 提供了一种方便、有效管理对话框创建、保存、回复的对话框机制，例如：onCreateDialog(int id)，onPrepareDialog(int id，Dialog dialog)，showDialog(int id)，dismissDialog(int id)等方法，在使用这些方法的时，Activity 将通过 getOwnerActivity()方法返回该 Activity 管理的对话框（Dialog）。这些方法的具体含义如表 7-3 所示。

表 7-3　Activity 管理对话框（Dialog）常用方法

方　　法	解 析 说 明
onCreateDialog(int　id)	当用户使用这个回调函数时，Android 系统会自动设置当前的 Activity 为该对话框的所有者，从而自动管理每一个对话框的状态并挂靠到 Activity 上，通过此方式为每一个对话框继承该 Activity 的特定属性
showDialog(int　id)	当用户想要显示一个对话框时，可以调用 showDialog(int id)方法，该方法需要传递一个整型的唯一标识列（ID）。当对话框第一次被请求时，Android 从所定义的 Activity 中调用 onCreateDialog(int　id)方法
onPrepareDialog(int id，　Dialog　dialog)	在对话框被显示之前，Android 还调用了可选的回调函数 onPrepareDialog(int id，　Dialog dialog)。如果用户想在每一次对话框被打开时改变其任何属性，用户可以调用该方法。这个方法需要传递一个对话框 ID 和一个在 onCreateDialog()中创建的对话框对象。onPrepareDialog(int id，Dialog dialog)和 onCreateDialog(int)的区别是：onPrepareDialog(int id，　Dialog dialog)这个方法在每次打开对话框时被调用，　而　onCreateDialog(int　id)　仅在对话框第一次打开时被调用。如果用户不定义 onPrepareDialog()，那么这个对话框将保持和上次打开时一样
dismissDialog(int　id)	当用户准备关闭对话框时，可以通过调用该对话框的 dismiss()来消除它。如果需要，用户还可以从这个 Activity 中调用 dismissDialog(int id)这个方法将根据用户传入的 ID，消除相应的对话框

接着具体讲解如何创建对话框 Dialog，Android 创建 Dialog 的方法有两种：

1）是通过 AlertDialog.Builder 初始化 Dialog，这种方式可以设置初始化的 Dialog，然后再调用 showDialog()方法，Builder 所提供的方法，如表 7-4 所示。

表 7-4　Dialog 的主要方法及其功能介绍

方　　法	功 能 说 明
setTitle()	为对话框设置 title（标题）
setIcon()	为对话框设置图标
setMessage()	为对话框设置提示信息
setItems()	为对话框设置要显示的一个普通 list（列表子项）一般用于显示几个命令时使用
setSingleChoiceItems()	为对话框设置显示一个单选的 List
setMultiChoiceItems()	为对话框设置显示一个复选框的 List
setPositiveButton()	为对话框添加一个“Yes”按钮
setNegativeButton()	为对话框添加一个“No”按钮

2）是通过将 androidManifest.xml 配置文件中的 Activity 属性设为 android:theme= "@android:style/Theme.Dialog，伪装为 Dialog。

下面通过一个示例讲述对话框 Dialog 的用法，示例的主要功能是提供各种消息对话框的显示，这些对话框在开发中常常使用到。主要有：普通消息框、确认框、输入框、单选框、多选框、列表框及图片框，示例代码运行结果如图 7-13 至图 7-20 所示。



图 7-13　各种对话框（Dialog）的使用示例代码运行结果（1）——初始化



图 7-14　各种对话框（Dialog）的使用示例代码运行结果（2）——普通消息框



图 7-15　各种对话框（Dialog）的使用示例代码运行结果（3）——确认框



图 7-16　各种对话框（Dialog）的使用示例代码运行结果（4）——输入消息框



图 7-17　各种对话框（Dialog）的使用示例代码运行结果（5）——单选框



图 7-18　各种对话框（Dialog）的使用示例代码运行结果（6）——多选消息框





图 7-19　各种对话框（Dialog）的使用示例代码运行结果（7）——列表消息框



图 7-20　各种对话框（Dialog）的使用示例代码运行结果（8）——图片框

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-18 至代码清单 7-20 所示。

代码清单 7-18　对话框Dialog使用示例（第 7 章\Demo_07_06）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
<Button
    android:id="@+id/btn_normal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="普通消息框"
    />

   <Button
    android:id="@+id/btn_confirm"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="确认框"
    />

    <Button
    android:id="@+id/btn_input"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="输入框"
    />

    <Button
    android:id="@+id/btn_singleChoice"
    android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        android:text="单选框"
        />

        <Button
        android:id="@+id/btn_multiChoice"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="多选框"
        />

        <Button
        android:id="@+id/btn_list"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="列表框"
        />

        <Button
        android:id="@+id/btn_pic"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="图片框"
        />
</LinearLayout>
```

代码清单 7-19　对话框Dialog使用示例（第 7 章\Demo_07_06）dialog.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:orientation="vertical">

    <TextView
        android:id="@+id/username"
        android:layout_height="wrap_content"
        android:layout_width="wrap_content"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
        android:text="账号"
        android:gravity="left"
        android:textAppearance="?android:attr/textAppearanceMedium" />

    <EditText
        android:id="@+id/username"
        android:layout_height="wrap_content"
        android:layout_width="fill_parent"
        android:layout_marginLeft="20dip"
        android:layout_marginRight="20dip"
```



```
            android:scrollHorizontally="true"
            android:autoText="false"
            android:capitalize="none"
            android:gravity="fill_horizontal"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <TextView
            android:id="@+id/password"
            android:layout_height="wrap_content"
            android:layout_width="wrap_content"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:text="密码"
            android:gravity="left"
            android:textAppearance="?android:attr/textAppearanceMedium" />

        <EditText
            android:id="@+id/password"
            android:layout_height="wrap_content"
            android:layout_width="fill_parent"
            android:layout_marginLeft="20dip"
            android:layout_marginRight="20dip"
            android:scrollHorizontally="true"
            android:autoText="false"
            android:capitalize="none"
            android:gravity="fill_horizontal"
            android:password="true"
            android:textAppearance="?android:attr/textAppearanceMedium" />
</LinearLayout>
```

代码清单 7-20　对话框Dialog使用示例（第 7 章\Demo_07_06\values）array.xml

```
<?xml version="1.0" encoding="utf-8"?>
<resources>
    <string-array name="hobby">
        <item>篮球</item>
        <item>足球</item>
        <item>排球</item>
    </string-array>
</resources>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 7-21 所示。

代码清单 7-21　对话框Dialog使用示例（第 7 章\Demo_07_06）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.app.AlertDialog;
import android.app.Dialog;
import android.app.ProgressDialog;
import android.content.DialogInterface;
```




```
import android.os.Bundle;
import android.view.LayoutInflater;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.ImageView;

/**
 * Demo_07_06 Dialog 的使用方法
 * @author
 *
 */

public class MainActivity extends Activity {

    private Button btn_normal;
    private Button btn_confirm;
    private Button btn_input;
    private Button btn_singleChoice;
    private Button btn_multiChoice;
    private Button btn_list;
    private Button btn_pic;

    // 定义一个 boolean 型数组（多选索引）
    boolean[] selected = new boolean[] { false, false, false };

    // 定义一个 int 型数组（单选、列表子项索引）
    int[] singeselected = new int[] { 0, 1, 2, 3 };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

         /*获取 Button 元素*/
        btn_normal = (Button) findViewById(R.id.btn_normal);
        btn_confirm = (Button) findViewById(R.id.btn_confirm);
        btn_input = (Button) findViewById(R.id.btn_input);
        btn_singleChoice = (Button) findViewById(R.id.btn_singleChoice);
        btn_multiChoice = (Button) findViewById(R.id.btn_multiChoice);
        btn_list = (Button) findViewById(R.id.btn_list);
        btn_pic = (Button) findViewById(R.id.btn_pic);

         /*监听 Button 事件*/
        btn_normal.setOnClickListener(normalClick);
        btn_confirm.setOnClickListener(confirm);
        btn_input.setOnClickListener(input);
        btn_singleChoice.setOnClickListener(singleChoice);
        btn_multiChoice.setOnClickListener(multiChoice);
        btn_list.setOnClickListener(list);
        btn_pic.setOnClickListener(pic);

    }
```



```
/**
 * 实现监听
 */
Button.OnClickListener normalClick = new Button.OnClickListener() {

    @Override
    public void onClick(View v) {
        // TODO Auto-generated method stub

    Dialog dialog = new AlertDialog.Builder(MainActivity.this)
            .setTitle("登录提示")
                // 设置标题
            .setMessage("这里需要登录！")
                // 设置内容
            .setPositiveButton("确定",
                new DialogInterface.OnClickListener() {
                    @Override
                    public void onClick(DialogInterface dialog,
                                    int whichButton) {
                                // 单击"确定"转向登录框

                    }
                })
           .setNeutralButton("退出",
                new DialogInterface.OnClickListener() {
                    @Override
                public void onClick(DialogInterface dialog,
                                int whichButton) {
                    // 单击"退出"按钮之后推出程序
                    MainActivity.this.finish();
                    }
                }).create();// 创建按钮

        // 显示对话框
        dialog.show();

    }

};

Button.OnClickListener confirm = new Button.OnClickListener() {

    ProgressDialog m_Dialog;

    @Override
    public void onClick(View v) {
      //读取布局 xml 文件
      LayoutInflater factory = LayoutInflater.from(MainActivity.this);
        // 得到自定义对话框
        final View DialogView = factory.inflate(R.layout.dialog, null);
        // 创建对话框
```

```
                    AlertDialog dlg = new AlertDialog.Builder(MainActivity.this)
                            .setTitle("登录框").setView(DialogView)//自定义对话框的样式
                            .setPositiveButton("确定",
                                new DialogInterface.OnClickListener() //设置事件监听
                                    {
                                        @Override
                                    public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                            // 输入完成后，单击"确定"开始登录
                                            m_Dialog = ProgressDialog.show(
                                                    MainActivity.this, "请等待...",
                                                    "正在为你登录...", true);

                                            new Thread() {
                                                @Override
                                                public void run() {
                                                    try {
                                                        sleep(3000);
                                                    } catch (Exception e) {
                                                        e.printStackTrace();
                                                    } finally {
                                                        // 取消 m_Dialog 对话框
                                                        m_Dialog.dismiss();
                                                    }
                                                }
                                            }.start();
                                        }
                                    })
                                .setNegativeButton("取消",
                                    new DialogInterface.OnClickListener() {
                                        @Override
                                        public void onClick(DialogInterface dialog,
                                                int whichButton) {
                                            // 单击"取消"按钮之后退出程序
                                            MainActivity.this.finish();
                                        }
                                    }).create();//创建
                    dlg.show();//显示

                }

            };
            Button.OnClickListener input = new Button.OnClickListener() {

                @Override
                public void onClick(View v) {

                    // TODO Auto-generated method stub
                    new AlertDialog.Builder(MainActivity.this)
                            .setTitle("请输入")
                            .setIcon(android.R.drawable.ic_dialog_info)
```



```
                        .setView(new EditText(MainActivity.this))
                        .setPositiveButton("确定",
                                new DialogInterface.OnClickListener() {
                                    @Override
                                    public void onClick(DialogInterface dialog,
                                            int whichButton) {

                                    }
                                })
                        .setNegativeButton("取消", null).show();
            }

    };

    Button.OnClickListener singleChoice = new Button.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub

            new AlertDialog.Builder(MainActivity.this)
                    .setTitle("请选择")
                    .setIcon(android.R.drawable.ic_dialog_info)
                    .setSingleChoiceItems(R.array.hobby, singeselected[0],
                            new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(DialogInterface dialog,
                                        int which) {
                                    String selectedStr = "";
                                    /* 循环遍历所选择的项 */
                                for (int i = 0; i < singeselected.length; i++) {
                                    if (singeselected[i] == which) {
                                        // 调用 values 中的 array.xml
                                        selectedStr = selectedStr
                                                + " "
                                                + getResources()
                                                .getStringArray(
                                                R.array.hobby)[i];
                                    }
                                }

                                /* 将内容显示在编辑本文框里 */
                                btn_singleChoice.setText(selectedStr);

                                dialog.dismiss();

                            }
                        })
                    .setNegativeButton("取消", null).show();
```




```
        }

    };

    Button.OnClickListener multiChoice = new Button.OnClickListener() {

        @Override
        public void onClick(View v) {
            // TODO Auto-generated method stub
            new AlertDialog.Builder(MainActivity.this)
                    .setTitle("多选框")
                    .setMultiChoiceItems(R.array.hobby, selected,
                            new DialogInterface.OnMultiChoiceClickListener() {

                                @Override
                                public void onClick(
                                        DialogInterface dialogInterface,
                                        int which, boolean isChecked) {
                                    selected[which] = isChecked;
                                }
                            })
                    .setPositiveButton("确定",
                            new DialogInterface.OnClickListener() {
                                @Override
                                public void onClick(
                                        DialogInterface dialogInterface,
                                        int which) {
                                String selectedStr = "";
                                /* 循环遍历所选择的项 */
                                for (int i = 0; i < selected.length; i++) {
                                        if (selected[i] == true) {
                                            selectedStr = selectedStr
                                                    + " "
                                                    + getResources()
                                                .getStringArray(
                                            // 调用 values 中的 array.xml
                                            R.array.hobby)[i];
                                        }
                                    }
                                    /* 将内容显示在编辑本文框里 */
                            btn_multiChoice.setText(selectedStr);
                                }
                            })
                    .setNegativeButton("取消", null)
                .setIcon(R.drawable.icon)
                .show();
        }

    };
```



```
Button.OnClickListener list = new Button.OnClickListener() {

        @Override
        public void onClick(View v) {
                // TODO Auto-generated method stub

                new AlertDialog.Builder(MainActivity.this)
                                                .setTitle("列表框")
                                                .setItems(R.array.hobby,listitem)
                                                .setNegativeButton("确定", null).show();
        }

};

/**
 * 列表框的子项
 */
DialogInterface.OnClickListener listitem =new DialogInterface.OnClickListener(){

        @Override
        public void onClick(DialogInterface dialog, int which) {
                // TODO Auto-generated method stub
                String selectedStr = "";
                /* 循环遍历所选择的项 */
        for (int i = 0; i < singeselected.length; i++) {
          if (singeselected[i] == which) {
                // 调用 values 中的 array.xml
                selectedStr = selectedStr
                                        + " "
                                                + getResources()
                                                .getStringArray(
                                                R.array.hobby)[i];
                        }
                }
                /* 将内容显示在编辑文本框里 */
                btn_list.setText(selectedStr);
                dialog.dismiss();

        }

};

Button.OnClickListener pic = new Button.OnClickListener() {

        @Override
        public void onClick(View v) {
                // TODO Auto-generated method stub
```

```
                        ImageView img = new ImageView(MainActivity.this);
                        img.setImageResource(R.drawable.icon);
                        new AlertDialog.Builder(MainActivity.this)
                                        .setTitle("图片框")
                                        .setView(img)
                                        .setPositiveButton("确定", null)
                                         .show();
                }
        };

}
```

7.1.7　进度条

进度条（ProgressBar）进度条有两种展示方式，一种是表盘形式（普通、小、大），另一种是条形填充形式。在 xml 布局中定义 ProgressBar 时，需要通过设置 style 属性来显示展示方式。

在可视确定性 ProgressBar 进度条中，可以设置两种类型的进度条。一种是用户操作的进度条，用来显示主进度；另一种是次要的进度条，主要用来显示中间进度，如在流媒体播放中处理缓冲区的进度。ProgressBar 进度条还可以是不确定性的进度条，在不确定模式下，进度条可循环显示动画，这种模式常用于应用程序执行任务的长度是未知的。ProgressBar 常用 xml 属性见表 7-5，重要方法及事件见表 7-6。

表 7-5　ProgressBar xml 属性

属　　性	功 能 说 明
android:progressBarStyle	默认进度条样式
android:progressBarStyleHorizontal	设置进度条水平样式
style	设置进度条显示方式，取值：?android:attr/progressBarStyleLarge 、progressBarStyle、progressBarStyleSmall 、progressBarStyleHorizontal
android:max	在 progressBarStyleHorizontal 方式下，设置进度条最大值
android:progress	在 progressBarStyleHorizontal 方式下，设置进度条主进度当前值
android:secondaryProgress	在 progressBarStyleHorizontal 方式下，设置进度条次进度当前值

表 7-6　ProgressBar 重要方法及事件

方　　法	功 能 说 明
getMax()	返回进度条的范围上限
getProgress()	返回主进度
getSecondaryProgress()	返回次要进度
incrementProgressBy(int diff)	指定增加的进度
isIndeterminate()	判断指示进度条是否在不确定模式下
setIndeterminate(boolean indeterminate)	设置进度条是否为不确定模式
setVisibility(int v)	设置进度条是否可见
onSizeChanged(int w，　int h，　int oldw，　int oldh)	进度条的进度值改变时触发此事件



RatingBar 是星式进度条，是特殊的一种进度条，常用属性见表 7-7。

表 7-7　RatingBar 常用属性

属　　性	功 能 说 明
android:isIndicator	设置是否是评级栏，起到指示器的作用。True 为指示器，用户不可进行操作
android:numStars	设置进度条的当前星数
android:stepSize	设置每次可等加的最小单位，值为浮点型

下面通过一个示例讲述进度条 ProgressBar、RatingBar 的用法，示例的主要功能是提供两种方式的进度条，条型进度条通过设置一个定时器，可动态加载进度；单击星型进度条可进行评级，示例代码运行结果如图 7-21 和图 7-22 所示。

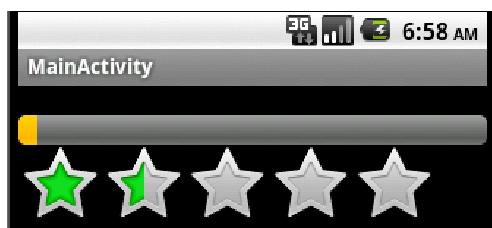


图 7-21　进度条（ProgressBar、RatingBar）示例代码运行结果（1）

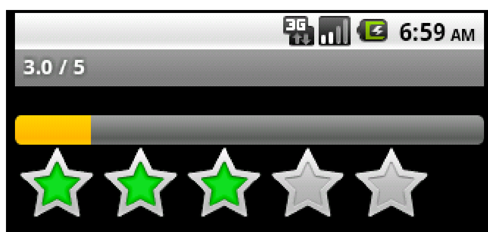


图 7-22　进度条（ProgressBar、RatingBar）示例代码运行结果（2）

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-22 所示。

代码清单 7-22　进度条（ProgressBar、RatingBar）示例（第 7 章\Demo_07_07）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
 <LinearLayout
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        >

        <ProgressBar
         android:id="@+id/progressbar"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:layout_marginTop="20dp"
         android:max="100"
         style="?android:attr/progressBarStyleHorizontal"
        />

        <RatingBar android:id="@+id/myRatingBar"
                    android:layout_width="wrap_content"
                  android:layout_height="wrap_content"
                  android:isIndicator="false"
```

```
            android:numStars="5"
            android:rating="1.5"
            android:stepSize="0.5"/>

</LinearLayout>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现这一效果，如代码清单 7-23 所示。

代码清单 7-23 进度条（ProgressBar、RatingBar）示例（第 7 章\Demo_07_07）MainActivity.java

```
package com.chen.android;

import java.util.Timer;
import java.util.TimerTask;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.widget.ProgressBar;
import android.widget.RatingBar;

/**
 * 进度条 ProgressBar
 *
 * @author
 *
 */
public class MainActivity extends Activity {
    //定义两种类型的进度条
    private ProgressBar progressBar;
    private RatingBar ratingBar;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取进度条对象
        progressBar = (ProgressBar) findViewById(R.id.progressbar);
        //获取星型进度条对象
        ratingBar = (RatingBar) findViewById(R.id.myRatingBar);
        // 实例化一个定时器
        Timer timer = new Timer();
        /*第一个参数为定时器，第二个参数为：从 0 开始，
        *第三个参数为：进度条的速度，越大越慢，越小越快*/
        timer.schedule(timerTask, 0, 500);
        /*为星型进度条设置监听事件*/
        ratingBar.setOnRatingBarChangeListener(onRatingBarChangeListener);

    }
```



```
        /* 开启一个线程 */
        private Handler handler = new Handler() {
            public void handleMessage(Message msg) {
                switch (msg.what) {
                case 1:
                    int currentProgress = progressBar.getProgress() + 2;
                    if (currentProgress > progressBar.getMax())
                        currentProgress = 0; // 断送进度条
                    progressBar.setProgress(currentProgress);// 显示进度条进度
                    break;
                }
                super.handleMessage(msg);
            }
        };
       /*设置一个定时器任务*/
        private TimerTask timerTask = new TimerTask() {
            // 信息
            @Override
            public void run() {
                Message message = new Message();
                message.what = 1;
                handler.sendMessage(message);
            }
        };

        /**
         * RatingBar 监听器
         */
        private RatingBar.OnRatingBarChangeListener onRatingBarChangeListener
                                 = new RatingBar.OnRatingBarChangeListener() {

            @Override
            public void onRatingChanged(RatingBar ratingBar, float rating,
                    boolean fromUser) {
                MainActivity.this
                        .setTitle(rating + " / " + ratingBar.getNumStars());
                Log.v("Rag", "RatingBar onRatingChanged， rating： " + rating
                        + ", fromUser: " + fromUser);
            }
        };

    }
```

7.2　重点剖析

在之前的章节中曾多次使用过线程，到底什么是线程，有什么作用？在程序中什么时候使用线程？怎么使用线程？本节将重点讲解线程的使用方法。

在多线程编程中，经常要使用 Handler，Thread、Runnable 和 HandlerThread 这几个类，它们之间有什么样的关系呢？首先介绍 Thread，它是 Android CPU 分配的最小单元。Handler 一般是在某个线程里创建的，相当于是一个线程的栈，数据的处理顺序是先进先出的处理顺序。且 Handler 和 Thread 是相互绑定的，一一对应的。Runnable 是一个接口，Thread 是 Runnable 的子类。所以说，这两者都算是同一个进程。而 HandlerThread 是可以处理消息循环的线程，他是一个拥有 Looper（循环）的线程，可以处理消息循环。

下面通过一个示例讲述 Thread、handler 和 Runnable 的用法，代码运行结果如图 7-23 所示。

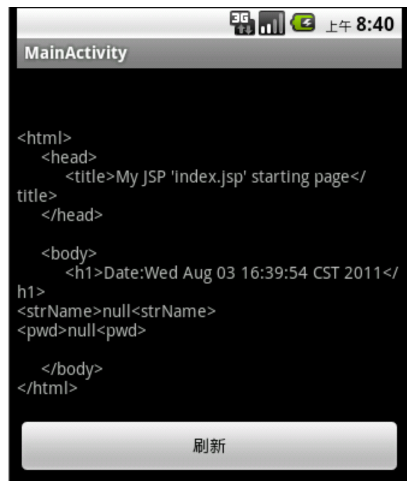


图 7-23　Thread、handler 和 Runnable 示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 7-24 所示。

代码清单 7-24　Thread、handler和Runnable示例（第 7 章\Demo_07_08）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
<TextView
      android:id="@+id/TextView01"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"
     />
  <Button
  android:id="@+id/Button01"
  android:layout_width="fill_parent"
  android:layout_height="wrap_content"
  android:text="刷新" />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现这一效果，如代码清单 7-25 所示。

代码清单 7-25　Thread、handler 和 Runnable 示例（第 7 章\Demo_07_08）MainActivity.java

```
package com.chen.android;

import java.io.BufferedReader;
import java.io.IOException;
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
```



```
import java.net.MalformedURLException;
import java.net.URL;
import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.util.Log;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
/**
 * Thread、handler 和 Runnable 的使用
 * @author
 *
 */
public class MainActivity extends Activity {
	//定义常量
	private final String DEBUG_TAG = "Activity02";
	//定义 TextView 对象
	private TextView mTextView;
	//定义 Button 对象
	private Button mButton;

	/** 主程序入口 */
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		//获取对象
		mTextView = (TextView) this.findViewById(R.id.TextView01);
		mButton = (Button) this.findViewById(R.id.Button01);
		//设置 button 按钮的监听
		mButton.setOnClickListener(new Button.OnClickListener() {
			@Override
			public void onClick(View arg0) {
				// TODO Auto-generated method stub
				refresh();
			}
		});
		// 开启线程实时刷新
		 new Thread(mRunnable).start();
	}

	// 刷新网页显示（每 5 秒刷新一次页面）
	private void refresh() {
	   //设置服务器连接地址
		String httpUrl = "http://192.168.1.178:8888/Test/index.jsp";
		String resultData = "";
		URL url = null;
		try {
```

```
                // 构造一个 URL 对象
                url = new URL(httpUrl);
            } catch (MalformedURLException e) {
                Log.e(DEBUG_TAG, "MalformedURLException");
            }
            if (url != null) {
                try {
                    // 使用 HttpURLConnection 打开连接
                    HttpURLConnection urlConn = (HttpURLConnection) url
                            .openConnection();
                    // 得到读取的内容(流)
                    InputStreamReader in = new InputStreamReader(
                            urlConn.getInputStream());
                    // 为输出创建 BufferedReader
                    BufferedReader buffer = new BufferedReader(in);
                    String inputLine = null;
                    // 使用循环来读取获得的数据
                    while (((inputLine = buffer.readLine()) != null)) {
                        // 我们在每一行后面加上一个“\n”来换行
                        resultData += inputLine + "\n";
                    }
                    // 关闭 InputStreamReader
                    in.close();
                    // 关闭 http 连接
                    urlConn.disconnect();
                    // 设置显示取得的内容
                    if (resultData != null) {
                        mTextView.setText(resultData);
                    } else {
                        mTextView.setText("读取的内容为 NULL");
                    }
                } catch (IOException e) {
                    e.toString();
                    Log.e(DEBUG_TAG, "IOException");
                }
            } else {
                Log.e(DEBUG_TAG, "Url NULL");
            }
        }
    //启动线程运行类
        private Runnable mRunnable = new Runnable() {
            public void run() {
                while (true) {
                    try {
                        // 每隔 5 秒执行一次（5000 毫秒）
                        Thread.sleep(5 * 1000);
                        // 发送消息
                        Message message = new Message();
                        //设置请求标志
                        message.what = 1;
```



```
                    mHandler.sendMessage(message);

                } catch (InterruptedException e) {
                    //  自动产生 catch 块
                    Log.e(DEBUG_TAG, e.toString());
                }
            }
        }
    };
    /*开启一个线程*/
    Handler mHandler = new Handler() {

        public void handleMessage(Message msg) {
            //  处理消息
            switch (msg.what) {
            case 1:
                refresh();
                break;

            }

        }
    };
}
```

最后，介绍 JSP 后台代码如何配合 Android 前端应用程序实现这一效果，如代码清单 7-26 所示。

代码清单 7-26　Thread、handler和Runnable示例（第 7 章\Demo_07_08）index.jsp

```
<%@ page language="java" import="java.util.*" pageEncoding="utf-8"%>
<%@page import="java.text.SimpleDateFormat"%>

<html>
    <head>
        <title>My JSP 'index.jsp' starting page</title>
    </head>

    <body>
        <%
            SimpleDateFormat fo = new SimpleDateFormat("yyyy-MM-dd HH:mm:ss");
            Date current = new Date();
            String str1 = fo.format(current);
            String str2 = current.toString();
            String name=request.getParameter("strName");
            String pwd=request.getParameter("strPwd");
            out.println("<h1>Date:" + str2 + "</h1>");
            out.println("<strName>"+name+"<strName>");
            out.println("<pwd>"+pwd+"<pwd>");
        %>
    </body>
</html>
```

7.3 查询功能实现

机票查询功能是本项目实现的难点，涉及的控件多且相对复杂。在项目中，将所讲述的基础知识配合起来使用，实现查票功能。

7.3.1 查询界面View实现

根据第 2 章手机订票系统实现效果图（图 2-7 至图 2-11）所示，查询功能的 View 实现采用 TabHost 布局，内嵌 LinearLayout、TabWidget 和 FrameLayout。与之前的线性布局不同，实现相对较复杂，这正是让读者不断巩固各种布局的使用，达到实战开发的目的。详见代码清单 7-27。

代码清单 7-27 查询界面View实现（第 7 章\ GanaSkyForTicketMain）ticketmain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<TabHost
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@android:id/tabhost"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <LinearLayout
        android:orientation="vertical"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent">
        <!-- 选项卡 -->
        <TabWidget
            android:id="@android:id/tabs"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content" />
        <FrameLayout
            android:id="@android:id/tabcontent"
            android:layout_width="fill_parent"
            android:layout_height="fill_parent">
            <!-- 选项卡一内容 -->
            <TableLayout
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:stretchColumns="1"
                android:id="@+id/table1">
                <TableRow>
                    <TextView
                        android:text="出发城市:"
                        android:layout_width="wrap_content"
                        android:layout_height="40sp"
                        android:padding="4dip"/>
                    <TextView
                        android:id="@+id/ticketmain_setCity"
```



```
            android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:padding="4dip"/>
        <ImageView
            android:id="@+id/ticketmain_imgSetCity"
            android:src="@drawable/checked"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:gravity="right"
             android:padding="4dip"/>
</TableRow>
<View
   android:layout_height="2dip"
   android:background="#FF909090" />
<TableRow
           android:layout_width="fill_parent"
             android:layout_height="wrap_content">
        <TextView
            android:text="到达城市:"
            android:padding="4dip"
             android:layout_width="wrap_content"
             android:layout_height="40sp" />
        <TextView
            android:id="@+id/ticketmain_offCity"
            android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:padding="4dip" />
        <ImageView
            android:id="@+id/ticketmain_imgoffCity"
            android:src="@drawable/checked"
             android:gravity="right"
             android:padding="4dip"
             android:layout_height="wrap_content" />
</TableRow>
<View
   android:layout_height="2dip"
   android:background="#FF909090" />
<TableRow>
        <TextView
            android:text="出发时间:"
            android:padding="4dip"
             android:layout_width="wrap_content"
             android:layout_height="40sp" />
        <TextView
            android:id="@+id/ticketmain_setTime"
            android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:padding="4dip" />
        <ImageView
            android:id="@+id/ticketmain_imgsetTime"
```




```
            android:src="@drawable/checked"
             android:gravity="right"
             android:padding="4dip"
             android:layout_height="wrap_content" />
</TableRow>
<View
     android:layout_height="2dip"
     android:background="#FF909090" />
<TableRow>
      <TextView
            android:text="仓位等级:"
            android:padding="3dip"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content" />
      <TextView
            android:id="@+id/ticketmain_setClass"
            android:text="经济舱"
             android:layout_width="80dip"
             android:layout_height="30dip"
             android:padding="3dip" />
      <Spinner
            android:id="@+id/ticketmain_spinnerSetClass"
            android:gravity="right"
             android:padding="3dip"
             android:layout_width="110dip"
             android:layout_height="wrap_content"
             android:layout_marginTop="5dip"
             android:layout_marginRight="5dip" />
</TableRow>
<View
     android:layout_height="2dip"
     android:background="#FF909090" />
<TableRow>
      <TextView
            android:text="航空公司:"
            android:padding="3dip"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content" />
      <TextView
            android:id="@+id/ticketmain_setTravelCompany"
            android:text="南方航空公司"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"
             android:padding="3dip" />
      <Spinner
            android:id="@+id/ticketmain_spinnerSetCompany"
             android:gravity="right"
             android:padding="3dip"
             android:layout_width="130dip"
             android:layout_height="wrap_content"
```



```
                    android:layout_marginTop="5dip"
                    android:layout_marginRight="5dip" />
            </TableRow>
            <View
                android:layout_height="2dip"
                android:background="#FF909090" />
            <TableRow
                android:layout_marginTop="5dip">
                <Button
                    android:id="@+id/ticketmain_btnSearchTicket"
                    android:text="查　询"
                    android:layout_span="2"
                    android:gravity="center"
                    android:paddingLeft="5dip"
                    android:layout_height="40dip"
                    android:layout_marginLeft="30dip"
                    >
                </Button>
                <Button
                    android:id="@+id/ticketmain_btnCancelSearchTicket"
                    android:text="取消"
                    android:gravity="center"
                    android:layout_column="2"
                    android:paddingLeft="5dip"
                    android:layout_height="40dip"
                    android:layout_marginLeft="20dip"
                    android:layout_marginRight="10dip" >
                </Button>
            </TableRow>
        </TableLayout>

        <!-- -第二页选项卡 -->
        <TableLayout
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:stretchColumns="1"
            android:id="@+id/table2">
            <TableRow>
                <TextView
                    android:text=" 出发城市:"
                    android:layout_width="80px"
                    android:layout_height="wrap_content" />
                <TextView
                    android:text="北京"
                    android:layout_width="80sp"
                    android:layout_height="wrap_content" />
                <ImageView
                    android:src="@drawable/checked"
                    android:layout_width="wrap_content" />
            </TableRow>
```

```
                    <View
                        android:layout_height="2dip"
                        android:background="#FF909090" />
                </TableLayout>
            </FrameLayout>

        </LinearLayout>
    </TabHost>
```

7.3.2　查询功能Control实现

查询功能的 Control 实现，就是机票查询功能的 Activity java 代码实现，实现过程如代码清单 7-28 所示。在实现过程中注意以下几点：

1）如何实现页面的切换及传值，在查询界面中单击城市选择按钮后跳转到城市选择界面，当选择了某一个城市后又切换回查票页面。

2）城市查询分出发城市和到达城市的查询，如何在一个页面中实现两个不同城市的选择与传值，这是本功能界面的重点之一。

3）为了减少用户再次操作，查询界面要提供“记忆”功能，查询了一次，下次再登录查询界面的时候，要把用户选择的信息呈现出来。

4）为了减少后台的压力，前台用户选择的信息也需要进行不为空的验证，增强平台的健壮性。

代码清单 7-28　查询界面 Control 实现（第 7 章\GanaSkyForTicketMain）TicketMainApp.java（详见本书源代码）

7.3.3　查询功能ASP实现

在查询界面中，因为机票查询需要读取 Eterm（中航信）的指令，查询需要等待一段时间，因此把组装的机票查询参数传到下一个页面，在信息列表界面组装参数与后台进行通信。因此，把与后台通信的代码放在信息列表中进行操作，在这里先给出查询的 ASP 代码，供下一章 Activity　java 代码在初始化时给列表控件赋数据源时调用。详见代码清单 7-29。

代码清单 7-29　查询功能ASP实现（第 7 章\GanaSkyForTicketMain）SearchTicketPage.aspx.cs（详见本书源代码）

城市选择功能执行 ASP 后台代码后返回的 xml 数据，如代码清单 7-30 所示。

代码清单 7-30　查询功能执行ASP后台代码后返回的xml数据示例（第 7 章\GanaSkyForTicketMain\assets\data）planeInfo.xml　（部分数据）

```
<?xml version="1.0" encoding="utf-8"?>
<FlightInfoInfo>
  <FlightInfo>
    <CompanyCode>HU</CompanyCode>
    <CompanyName>海南航空</CompanyName>
    <FlightNumber>HU7147</FlightNumber>
    <PlaneModel>330 </PlaneModel>
    <TakeOff>2011/6/3 7:55:00</TakeOff>
```



```
<Landing>2011/6/3 10:45:00</Landing>
<FromAirportCode>PEK</FromAirportCode>
<FromAirportName>北京</FromAirportName>
<FormatedFromAiport>北京(PEK)</FormatedFromAiport>
<LandingAirportCode>CTU</LandingAirportCode>
<LandingAirportName>成都</LandingAirportName>
<FormatedLandingAirport>成都(CTU)</FormatedLandingAirport>
<IsPassBy>False</IsPassBy>
<IsElectronicTicket>True</IsElectronicTicket>
<AirportConstructionFee>50</AirportConstructionFee>
<FuelSurcharge>110</FuelSurcharge>
<Distance>1697</Distance>
<Classes></Classes>
<AvailableClassCount>6</AvailableClassCount>
<AvailableClassPolicyCount>19</AvailableClassPolicyCount>
<xmlClasses>
  <ClassInfoInfo>
    <ClassInfo>
      <ClassCode>H</ClassCode>
      <ClassName>85 折</ClassName>
      <AvailableSeat>≥ 9</AvailableSeat>
      <PublicPrice>1220</PublicPrice>
      <RefundPercent>0</RefundPercent>
      <SettlementPrice>1220</SettlementPrice>
      <IsPolicyAvailable>True</IsPolicyAvailable>
    </ClassInfo>
    <ClassInfo>
      <ClassCode>B</ClassCode>
      <ClassName>90 折</ClassName>
      <AvailableSeat>≥ 9</AvailableSeat>
      <PublicPrice>1300</PublicPrice>
      <RefundPercent>0</RefundPercent>
      <SettlementPrice>1300</SettlementPrice>
      <IsPolicyAvailable>True</IsPolicyAvailable>
    </ClassInfo>
    <ClassInfo>
      <ClassCode>Y</ClassCode>
      <ClassName>经济舱</ClassName>
      <AvailableSeat>≥ 9</AvailableSeat>
      <PublicPrice>1440</PublicPrice>
      <RefundPercent>0</RefundPercent>
      <SettlementPrice>1440</SettlementPrice>
      <IsPolicyAvailable>True</IsPolicyAvailable>
    </ClassInfo>
    <ClassInfo>
      <ClassCode>F</ClassCode>
      <ClassName>头等舱</ClassName>
      <AvailableSeat>≥ 9</AvailableSeat>
      <PublicPrice>2880</PublicPrice>
      <RefundPercent>0</RefundPercent>
```




```
                <SettlementPrice>2880</SettlementPrice>
                <IsPolicyAvailable>True</IsPolicyAvailable>
            </ClassInfo>
        </ClassInfoInfo>
    </xmlClasses>
</FlightInfo>
</FlightInfoInfo>
```

第 8 章　信息列表功能实现

本章是对第 7 章基础控件的巩固，没有涉及新的控件学习，涉及的知识重点有两个：

1）如何对日期的各种形式进行操作，包括对当前和具体日期不同格式的处理和转换。

2）进一步学习更为复杂的 ListView 数据源及其数据子项的使用。

8.1　重点剖析

本节介绍功能实现中需要重点掌握的 Android 编程方法与编程思路介绍日期格式处理与转换和带图片多行 ListView 子项的实现内容。

8.1.1　日期格式处理与转换

在开发过程中，日期的显示格式存在多种形式。例如有 2011-8-3 方式、2011/8/3 方式、2011-8-3 8：30 方式、2011/8/3 8：30 方式，还有只需要显示时间的 8：30 方式等。在 ASP 后台返回的时间格式是 2011/8/3 8：30，需要使用 replaceAll("/","-")方法进行日期格式的简单转换。在开发中，还需要根据给定的具体时间得到当天是星期几，这就需要进行相应日期处理与转换。

下面通过一个示例讲述日期格式转换的用法，示例的主要功能是输入一个日期（需要定义格式）从而对日期进行多种格式转换并计算出当天是星期几，示例代码运行结果如图 8−1 和图 8−2 所示。

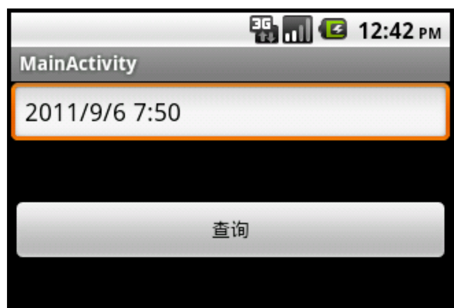


图 8−1　日期格式处理与转换示例（1）——初始化



图 8−2　日期格式处理与转换示例（2）——日期转换与换算

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 8−1 所示。

代码清单 8−1　日期格式处理与转换示例（第 8 章\Demo_08_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
```

```
xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

        <EditText
                android:id="@+id/ed_Date"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:text="2011/9/6 7:50"
         />
        <TextView
               android:id="@+id/tv_Date"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content" >
         </TextView>
           <TextView
               android:id="@+id/tv_DateTime"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content" >
         </TextView>
        <Button
                android:id="@+id/btn_Search"
                android:text="查询"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content" >
         </Button>

</LinearLayout>
```

其次，介绍 Activity java 代码如何配合 xml 布局实现日期格式处理与转换这一功能，如代码清单 8-2 所示。

代码清单 8-2　日期格式处理与转换示例（第 8 章\Demo_08_01）MainActivity.java

```
package com.chen.android;

import java.text.SimpleDateFormat;
import java.util.Date;
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.TextView;
import android.widget.Toast;
/**
 * 日期转换与换算
 * @author
```



```
 *
 */
public class MainActivity extends Activity {
	/*  声明对象  */
	private TextView tv_Date;
	private TextView tv_DateTime;
	private EditText ed_Date;
	private Button btn_Search;
	private String strDate;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		/*  获取对象  */
		tv_Date = (TextView) findViewById(R.id.tv_Date);
		tv_DateTime= (TextView) findViewById(R.id.tv_DateTime);
		ed_Date = (EditText) findViewById(R.id.ed_Date);
		btn_Search = (Button) findViewById(R.id.btn_Search);

		/*  设置监听  */
		btn_Search.setOnClickListener(onSearch);
	}

	/**
	 *  方法实现
	 */
	Button.OnClickListener onSearch = new Button.OnClickListener() {

		@Override
		public void onClick(View v) {
			strDate = ed_Date.getText().toString();
			/*字符串转换*/
			strDate=strDate.replaceAll("/", "-");
			/*转换输出*/
			tv_Date.setText("转换为："+strDate);
			/*截取时间*/
			tv_DateTime.setText("截取时间："+getDateTime(strDate));
			/*换算当天星期几*/
			strDate = getWeekDay(strDate);
			/*星期输入*/
			DisplayToast(ed_Date.getText().toString()+"是："  + strDate);
		}

	};

	/**
	 *  得到星期
	 */
```

```
private String getWeekDay(String strDate) {
        String returndate = "星期";
        int weekDay = 0;
        if (strDate != null) {
                //这里定义输入的字符串的格式，传入的是 yyyy-MM-dd hh:mm 格式
                SimpleDateFormat simpleDateFormat = new SimpleDateFormat(
                                "yyyy-MM-dd hh:mm");
                Date date = new Date();
                try {
                        date = simpleDateFormat.parse(strDate);
                        weekDay = date.getDay();
                        switch (weekDay) {
                        case 0:
                                returndate += "日";
                                break;
                        case 1:
                                returndate += "一";
                                break;
                        case 2:
                                returndate += "二";
                                break;
                        case 3:
                                returndate += "三";
                                break;
                        case 4:
                                returndate += "四";
                                break;
                        case 5:
                                returndate += "五";
                                break;
                        case 6:
                                returndate += "六";
                                break;
                        default:
                                break;
                        }
                } catch (java.text.ParseException e) {
                        e.printStackTrace();
                }

        }
        return returndate;
}

/**
 * 得到时间（小时和分钟）
 */
private String getDateTime(String strDate) {
        String returndate = "";
        if (strDate != null) {
```



```
				//这里定义输入的字符串的格式，传入的是 yyyy-MM-dd hh:mm 格式
				SimpleDateFormat simpleDateFormat = new SimpleDateFormat(
						"yyyy-MM-dd hh:mm");
				Date date = new Date();
				try {
					date = simpleDateFormat.parse(strDate);
					returndate = String.valueOf(date.getHours() + ":"
							+ date.getMinutes());
				} catch (java.text.ParseException e) {
					e.printStackTrace();
				}
			}
			return returndate;
		}

		/**
		 * Toast 提示框
		 */
		public void DisplayToast(String str) {
			Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
			// 设置 toast 显示的位置
			toast.setGravity(Gravity.TOP, 0, 220);
			// 显示该 toast
			toast.show();
		}
	}
```

8.1.2　带图片多行ListView子项

在本书中，讲解了各种形式 ListView 的使用，足以见其在 Android 应用程序开发中使用频繁且重要，证实了 ListView 的子项在开发过程中具有常用性、多变性与复杂性等特点，只有通过不断的深入训练才能攻克这一知识重点。本示例的主要功能是读取手机应用程序的图标、程序名及使用到的包，作为 ListView 的数据源，并显示在 ListView 控件上，代码运行结果如图 8-3 所示。

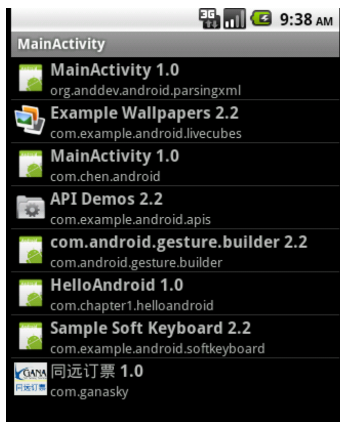


图 8-3　带图片多行 ListView 子项示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 8-3、8-4 所示。

代码清单 8-3　带图片多行ListView子项示例（第 8 章\Demo_08_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content">

     <ListView
         android:id="@+id/list_view"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content" />

</LinearLayout>
```

代码清单 8-4　带图片多行ListView子项示例（第 8 章\Demo_08_02）list_item.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:orientation="horizontal">

     <ImageView
         android:id="@+id/app_icon"
         android:layout_gravity="center_vertical"
         android:layout_width="32.0dip"
         android:layout_height="32.0dip"
         android:layout_marginLeft="3.0dip"
         android:layout_marginRight="3.0dip" />
     <LinearLayout
         android:orientation="vertical"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:layout_weight="1.0">
         <TextView
             android:id="@+id/app_title"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:textSize="16.0dip"
             android:textStyle="bold" />
         <TextView
             android:id="@+id/app_package"
             android:layout_width="fill_parent"
             android:layout_height="wrap_content"
             android:textSize="13.0dip" />
     </LinearLayout>
```



```
</LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现带图片多行 ListView 显示子项这一效果，如代码清单 8-5、8-6 所示。

代码清单 8-5 带图片多行ListView子项示例（第 8 章\Demo_08_02）MainActivity.java

```
package com.chen.android;

import java.util.ArrayList;
import java.util.HashMap;
import java.util.List;
import android.app.Activity;
import android.content.pm.ApplicationInfo;
import android.content.pm.PackageInfo;
import android.graphics.drawable.Drawable;
import android.os.Bundle;
import android.widget.ListView;
import android.widget.SimpleAdapter;
/**
 * 带图片多行 List 子项读取手机安装的程序的图片和程序名
 * @author
 *
 */
public class MainActivity extends Activity {
    private static final String APP_ICON = "app_icon";
    private static final String APP_TITLE = "app_title";
    private static final String APP_PACKAGE = "app_package";

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
       /*获取 ListView 对象*/
        ListView listView = (ListView)findViewById(R.id.list_view);
        /*SimpleAdapter 适配器加载资源到 list_item 中*/
        SimpleAdapter adapter = new SimpleIconAdapter(this,
                getInstalledApps(false),
                R.layout.list_item,
                new String[] {APP_ICON, APP_TITLE, APP_PACKAGE},
                new int[] {R.id.app_icon, R.id.app_title, R.id.app_package});
        listView.setAdapter(adapter);
    }
  /**
   * 获得手机已安装程序的信息
   */
    private List<HashMap<String, Object>>
                                getInstalledApps(boolean getSysPackages) {
        List<HashMap<String, Object>> listItem
                                = new ArrayList<HashMap<String, Object>>();
```




```
                /*获取所安装程序的文件包*/
                List<PackageInfo> pkgs = getPackageManager().getInstalledPackages(0);
                for (int i = 0; i< pkgs.size(); i++) {
                    PackageInfo pkg = pkgs.get(i);
                if (!getSysPackages && (pkg.applicationInfo.flags
                                            & ApplicationInfo.FLAG_SYSTEM) > 0) {
                        continue;
                    }
                    /*获取所安装程序的图片*/
                    Drawable icon = pkg.applicationInfo.loadIcon(getPackageManager());
                    /*获取所安装的程序名*/
                    String label = pkg.applicationInfo.loadLabel(getPackageManager()).toString();
                    /*获取所安装程序的包的版本名称*/
                    String version = pkg.versionName;
                    /*获取所安装程序的包名*/
                    String packageName = pkg.packageName;
                   /*通过 HashMap（Hash map）组装数据*/
                    HashMap<String, Object> map = new HashMap<String, Object>();
                    map.put(APP_ICON, icon);
                    map.put(APP_TITLE, label + " " + version);
                    map.put(APP_PACKAGE, packageName);
                    listItem.add(map);
                }
                return listItem;
            }
        }
```

代码清单 8-6　带图片多行ListView子项示例（第 8 章\Demo_08_02）SimpleIconAdapter.java

```
        package com.chen.android;
        import java.util.List;
        import java.util.Map;
        import android.content.Context;
        import android.graphics.drawable.Drawable;
        import android.view.View;
        import android.view.ViewGroup;
        import android.widget.ImageView;
        import android.widget.SimpleAdapter;
        /**
          * 数据源适配器
          * @author
          *
          */
        public class SimpleIconAdapter extends SimpleAdapter {

            public SimpleIconAdapter(Context context, List<? extends Map<String, ?>> data,
                                                    int resource, String[] from, int[] to) {
                super(context, data, resource, from, to);
            }
```



```
        @SuppressWarnings("rawtypes")
        @Override
        public View getView(int position, View convertView, ViewGroup parent) {

            View view = super.getView(position, convertView, parent);
         /*配置显示的图片*/
            ImageView iv = (ImageView)view.findViewById(R.id.app_icon);
            iv.setImageDrawable((Drawable)((Map)getItem(position)).get("app_icon"));
            return view;
        }
    }
```

8.2　信息列表功能实现

手机订票系统的信息列表功能，就是将查询到的航空信息通过 ListView 列表显示出来，包括显示各个航班及舱位的各种重要信息。实现的重点是如何组装查票信息与 ASP 后台程序进行通信，接收返回的航班信息并进行相应的数据解析。

8.2.1　信息列表功能View实现

如第 2 章手机订票系统实现效果图（图 2-12）所示，信息列表功能的 View 实现采用 LinearLayout 布局，内嵌 RelativeLayout 布局的形式。在 ListView 中要嵌 flightlist_items.xml 子项，实现起来也相对复杂，只有通过不断的巩固和训练才能使个人的 Android 开发技术更上一层楼。详见代码清单 8-7、8-8。

代码清单 8-7　信息列表功能View实现（第 8 章\GanaSkyForFlightInfoMain）flightlistmain.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    android:id="@+id/LinearLayout01"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:background="@drawable/buttonbgon"
    >
    <RelativeLayout
        android:layout_width="wrap_content"
        android:layout_height="wrap_content" >
       <TextView
           android:id="@+id/flightlistmain_txtTime"
           android:layout_width="wrap_content"
          android:layout_height="wrap_content"
             android:layout_marginLeft="5dip"
             android:layout_centerVertical="true"/>
```




```
        <TextView
            android:id="@+id/flightlistmain_txtSetCity"
            android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginLeft="30dip"
          android:layout_toRightOf="@+id/flightlistmain_txtTime"
          android:layout_alignTop="@+id/flightlistmain_txtTime" />
      <TextView
           android:id="@+id/flightlistmain_arrow"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:layout_marginLeft="5dip"
              android:layout_toRightOf="@+id/flightlistmain_txtSetCity"
              android:layout_alignTop="@+id/flightlistmain_txtSetCity" />

      <TextView
            android:id="@+id/flightlistmain_txtOffCity"
               android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginLeft="5dip"
         android:layout_toRightOf="@+id/flightlistmain_arrow"
         android:layout_alignTop="@+id/flightlistmain_arrow" />

   </RelativeLayout>

  <RelativeLayout
            android:layout_width="wrap_content"
               android:layout_height="wrap_content" >
              <Button
                 android:id="@+id/flightlistmain_orderPrice"
                 android:layout_width="80dip"
                 android:layout_height="40dip"
                 android:layout_alignParentLeft="true"
                 android:layout_marginLeft="5dip"
                 android:text="价格排序"
                 />
             <Button
                 android:id="@+id/flightlistmain_orderTime"
                 android:layout_width="80dip"
                 android:layout_height="40dip"
                 android:layout_marginLeft="10dip"
                 android:layout_toRightOf="@+id/flightlistmain_orderPrice"
                 android:layout_alignTop="@+id/flightlistmain_orderPrice"
                 android:text="时间排序"
                 />
              <TextView
            android:id="@+id/flightlistmain_txtflightCount"
               android:layout_width="wrap_content"
         android:layout_height="wrap_content"
         android:layout_marginLeft="5dip"
```



```
                android:visibility="invisible"
                android:layout_toRightOf="@+id/flightlistmain_orderTime"
                android:layout_alignTop="@+id/flightlistmain_orderTime" />

        </RelativeLayout>

          <!-- 需要不断循环的子项 -->
        <RelativeLayout
             android:layout_width="fill_parent"
              android:layout_height="wrap_content" >
              <ListView
                   android:layout_width="wrap_content"
                    android:layout_height="wrap_content"
                    android:id="@+id/flightlistmain_listFlight" />
        </RelativeLayout>

    </LinearLayout>
```

代码清单 8-8　信息列表功能View实现（第 8 章\GanaSkyForFlightInfoMain）flightlist_items.xml

```
    <?xml version="1.0" encoding="utf-8"?>
    <LinearLayout android:id="@+id/fligtLisy_items"
         xmlns:android="http://schemas.android.com/apk/res/android"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:orientation="horizontal">
          <!-- -图片和其他三项水平排列 -->
          <ImageView
                 android:id="@+id/flightList_CompanyCode"
                 android:layout_gravity="center_vertical"
                 android:layout_height="wrap_content"
                 android:layout_width="wrap_content"
                 android:layout_marginRight="1.0dip" />
          <LinearLayout
              android:orientation="vertical"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_weight="1.0">
                 <!-- 第一行 -->
                 <LinearLayout
                         android:orientation="horizontal"
                         android:layout_width="fill_parent"
                         android:layout_height="wrap_content"
                         android:layout_weight="1.0">
                         <TextView
                                android:id="@+id/flightList_FlightNumber"
                                android:layout_height="wrap_content"
                                 android:layout_width="wrap_content"
                                 android:layout_centerVertical="true" />
```




```
                <TextView
                    android:id="@+id/flightList_ClassName"
                    android:layout_height="wrap_content"
                     android:layout_width="wrap_content"
                     android:layout_marginLeft="10dip"
                     android:layout_centerVertical="true" />
                <TextView
                    android:id="@+id/flightList_publicPriceText"
                    android:layout_height="wrap_content"
                     android:layout_width="wrap_content"
                     android:layout_marginLeft="40dip"
                     android:layout_centerVertical="true" />
                <TextView
                    android:id="@+id/flightList_PublicPrice"
                    android:layout_height="wrap_content"
                     android:layout_width="wrap_content"
                     android:textColor="#00ff00"
                     android:textStyle="bold"
                     android:layout_centerVertical="true" />

        </LinearLayout>
         <!-- 第二行 -->
         <LinearLayout
               android:orientation="horizontal"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:layout_weight="1.0">
               <TextView
                    android:id="@+id/flightList_CompanyName"
                    android:layout_height="wrap_content"
                     android:layout_width="wrap_content"
                     android:layout_centerVertical="true" />
                <TextView
                    android:id="@+id/flightList_FromAirportName"
                    android:layout_height="wrap_content"
                     android:layout_width="wrap_content"
                     android:layout_marginLeft="10dip"
                     android:layout_centerVertical="true" />
                <!-- 航班起飞的时间 -->
                <TextView
                    android:id="@+id/flightList_TakeOff"
                    android:layout_height="wrap_content"
                     android:layout_width="wrap_content"
                     android:layout_marginLeft="2dip"
                     android:layout_centerVertical="true" />
                <!-- 机票数 -->
                <TextView
                    android:id="@+id/flightList_AvailableSeat"
                    android:layout_height="wrap_content"
                     android:layout_width="wrap_content"
```



```
                                android:layout_marginLeft="10dip"
                                android:layout_centerVertical="true" />
                           <TextView
                               android:id="@+id/flightList_AvailableSeat2"
                               android:layout_height="wrap_content"
                                android:layout_width="wrap_content"
                                android:layout_centerVertical="true" />
                </LinearLayout>
                    <!-- 第三行 -->
                  <LinearLayout
                          android:orientation="horizontal"
                          android:layout_width="fill_parent"
                          android:layout_height="wrap_content"
                          android:layout_weight="1.0">
                          <TextView
                               android:id="@+id/flightList_LandingAirportName"
                               android:layout_height="wrap_content"
                                android:layout_width="wrap_content"
                                android:layout_marginLeft="65dip"
                                android:layout_centerVertical="true" />
                           <!-- 航班降落的时间 -->
                           <TextView
                               android:id="@+id/flightList_Landing"
                               android:layout_height="wrap_content"
                                android:layout_width="wrap_content"
                                android:layout_marginLeft="2dip"
                                android:layout_centerVertical="true" />

                </LinearLayout>

          </LinearLayout>

</LinearLayout>
```

8.2.2　信息列表功能 Model类实现

信息列表页面涉及航班信息和舱位信息的组装，因此需要建这两个 Model 类（FlightInfo.java 和 ClassInfo.java）。 见代码清单 8-9、8-10。

代码清单 8-9　信息列表功能Model类实现（第 8 章\GanaSkyForFlightInfoMain）FlightInfo.java

```
package com.ganasky.model;

import java.util.List;
/**
 * 航班信息类
 * @author
 *
 */
```




```
public class FlightInfo {

    private String CompanyCode;
    private String CompanyName;
    private String FlightNumber;
    private String PlaneModel;
    private String TakeOff;
    private String Landing;
    private String FromAirportCode;
    private String FromAirportName;
    private String FormatedFromAiport;
    private String LandingAirportCode;
    private String LandingAirportName;
    private String FormatedLandingAirport;
    private String IsPassBy;
    private String IsElectronicTicket;
    private String AirportConstructionFee;
    private String FuelSurcharge;
    private String Distance;
    private String AvailableClassCount;
    private String AvailableClassPolicyCount;
    private List<ClassInfo> classInfo;
    //日期
    private String DateTime;
    //星期
    private String weekDay;

    public String getCompanyCode() {
        return CompanyCode;
    }
    public void setCompanyCode(String companyCode) {
        CompanyCode = companyCode;
    }
    public String getCompanyName() {
        return CompanyName;
    }
    public void setCompanyName(String companyName) {
        CompanyName = companyName;
    }
    public String getFlightNumber() {
        return FlightNumber;
    }
    public void setFlightNumber(String flightNumber) {
        FlightNumber = flightNumber;
    }
    public String getPlaneModel() {
        return PlaneModel;
    }
    public void setPlaneModel(String planeModel) {
        PlaneModel = planeModel;
```



```
}
public String getTakeOff() {
    return TakeOff;
}
public void setTakeOff(String takeOff) {
    TakeOff = takeOff;
}
public String getLanding() {
    return Landing;
}
public void setLanding(String landing) {
    Landing = landing;
}
public String getFromAirportCode() {
    return FromAirportCode;
}
public void setFromAirportCode(String fromAirportCode) {
    FromAirportCode = fromAirportCode;
}
public String getFromAirportName() {
    return FromAirportName;
}
public void setFromAirportName(String fromAirportName) {
    FromAirportName = fromAirportName;
}
public String getFormatedFromAiport() {
    return FormatedFromAiport;
}
public void setFormatedFromAiport(String formatedFromAiport) {
    FormatedFromAiport = formatedFromAiport;
}
public String getLandingAirportCode() {
    return LandingAirportCode;
}
public void setLandingAirportCode(String landingAirportCode) {
    LandingAirportCode = landingAirportCode;
}
public String getLandingAirportName() {
    return LandingAirportName;
}
public void setLandingAirportName(String landingAirportName) {
    LandingAirportName = landingAirportName;
}
public String getFormatedLandingAirport() {
    return FormatedLandingAirport;
}
public void setFormatedLandingAirport(String formatedLandingAirport) {
    FormatedLandingAirport = formatedLandingAirport;
}
public String getIsPassBy() {
```




```
        return IsPassBy;
    }
    public void setIsPassBy(String isPassBy) {
        IsPassBy = isPassBy;
    }
    public String getIsElectronicTicket() {
        return IsElectronicTicket;
    }
    public void setIsElectronicTicket(String isElectronicTicket) {
        IsElectronicTicket = isElectronicTicket;
    }
    public String getAirportConstructionFee() {
        return AirportConstructionFee;
    }
    public void setAirportConstructionFee(String airportConstructionFee) {
        AirportConstructionFee = airportConstructionFee;
    }
    public String getFuelSurcharge() {
        return FuelSurcharge;
    }
    public void setFuelSurcharge(String fuelSurcharge) {
        FuelSurcharge = fuelSurcharge;
    }
    public String getDistance() {
        return Distance;
    }
    public void setDistance(String distance) {
        Distance = distance;
    }
    public String getAvailableClassCount() {
        return AvailableClassCount;
    }
    public void setAvailableClassCount(String availableClassCount) {
        AvailableClassCount = availableClassCount;
    }
    public String getAvailableClassPolicyCount() {
        return AvailableClassPolicyCount;
    }
    public void setAvailableClassPolicyCount(String availableClassPolicyCount) {
        AvailableClassPolicyCount = availableClassPolicyCount;
    }
    public List<ClassInfo> getClassInfo() {
        return classInfo;
    }
    public void setClassInfo(List<ClassInfo> classInfo) {
        this.classInfo = classInfo;
    }
    public String getDateTime() {
        return DateTime;
    }
```



```
        public void setDateTime(String dateTime) {
            DateTime = dateTime;
        }
        public String getWeekDay() {
            return weekDay;
        }
        public void setWeekDay(String weekDay) {
            this.weekDay = weekDay;
        }

    }
```

代码清单 8-10　信息列表功能 Model类实现（第 8 章\GanaSkyForFlightInfoMain）ClassInfo.java

```
    package com.ganasky.model;
    /**
     * 航班仓位类
     * @author
     *
     */
    public class ClassInfo {
        /*定义属性*/
        private String ClassCode;
        private String ClassName;
        private String AvailableSeat;
        private String PublicPrice;
        private String RefundPercent;
        private String SettlementPrice;
        private String IsPolicyAvailable;

        /*封装属性(Get/Set)*/
        public String getClassCode() {
            return ClassCode;
        }
        public void setClassCode(String classCode) {
            ClassCode = classCode;
        }
        public String getClassName() {
            return ClassName;
        }
        public void setClassName(String className) {
            ClassName = className;
        }
        public String getAvailableSeat() {
            return AvailableSeat;
        }
        public void setAvailableSeat(String availableSeat) {
            AvailableSeat = availableSeat;
        }
```

```
        public String getPublicPrice() {
            return PublicPrice;
        }
        public void setPublicPrice(String publicPrice) {
            PublicPrice = publicPrice;
        }
        public String getRefundPercent() {
            return RefundPercent;
        }
        public void setRefundPercent(String refundPercent) {
            RefundPercent = refundPercent;
        }
        public String getSettlementPrice() {
            return SettlementPrice;
        }
        public void setSettlementPrice(String settlementPrice) {
            SettlementPrice = settlementPrice;
        }
        public String getIsPolicyAvailable() {
            return IsPolicyAvailable;
        }
        public void setIsPolicyAvailable(String isPolicyAvailable) {
            IsPolicyAvailable = isPolicyAvailable;
        }

    }
```

8.2.3 信息列表功能Control实现

信息列表功能 Control 实现，就是航空信息列表 Activity java 代码实现，如代码清单 8-11、8-12 所示。在实现过程中需要注意的有以下几点：

1）在初始化控件时，就要根据上一个页面（TicketMainApp.java）传入的参数，进行相应的组装，并与 ASP 后台进行通信。在开发过程中，通信会遇到一些数据流的问题，如数据读取中断的问题，这需要通过设置延迟连接时长来解决。使用之前在帮助类中创建好的 getLongPostHttpURLConnByUrl()方法。

2）要正确解析通信过程中返回的航班信息，本例还是使用 DOM 方式解析 XML 数据，读者可采用之前学习过的其他方式解析 XML。

3）整个航班数据信息的流向是：将航班信息逐条解析并保存在 FlightInfo.java 和 ClassInfo.java 类中，然后再批量保存到 List<FlightInfo>、List<ClassInfo>中，接着又解析 List<FlightInfo>中的数据保存到 ArrayList<HashMap<String，Object>> listItem 数组列表中，最后作为 ListView 的子项数据源传入并显示在界面上。

4）ListView 控件的子项显示相对复杂，如果要为子项增加图片或动态添加文字，需要在子项适配器 getView(int position，View convertView，ViewGroup parent)方法中进行相应操作。

5）如何读取 ListView 子项的值，需要监听 ListView.setOnItemClickListener()事件，实现 onItemClick()方法，调用 ListView.getItemAtPosition(position)对象，取得子项 map 的值，然后



将值保存在 FlightInfo.java 和 ClassInfo.java 类中。

6）由于机票信息量较多，我们考虑按照价格或时间排序，这需要使用 Comparator 比较器，可以调用需要比较字段的 compareTo()方法，然后再使用 Collections.sort()方法对数据源进行排序。

代码清单 8-11　信息列表功能Control实现（第 8 章\ GanaSkyForFlightInfoMain）FlightInfoMainApp.java（详见本书源代码）

代码清单 8-12　信息列表功能Control实现（第 8 章\GanaSkyForFlightInfoMain）ImageAdapterOfFlightList.java（详见本书源代码）

第 9 章　List详情功能实现

第 7、8 章涉及的知识点较多，且是手机订票系统这一项目在开发过程中的难点。本章的内容相对简单，主要是显示信息列表的详情。可以在信息列表界面中加入文字交换器和图片交换器控件，动态显示文字和图片。每一个列表子项的信息在上一个界面中显示相对有限，为了让用户更多地了解列表信息，应该提供信息列表详情功能，将上一章订票系统中读取到的航班信息充分展示出来。

本章中基础控件部分主要学习 TextSwitcher 与 ImageSwitcher，这两者用法类似。只不过 ImageSwitcher 规定的 View 为 ImageView，而 TextSwitcher 为 TextView。实现不同的 View，需要在 public View makeView()这个方法里设置。若是 ImageSwitcher 则在 makeView()里定义 ImageView 并将其返回给 ImageSwitcher， 而 TextSwitcher 则在 makeView 里定义 TextView 并将其返回给 TextSwitcher。

细心的读者会发现，本书的章节排列顺序基本上遵循一种循环渐进、温故而知新的过程，当某一章出现较多基础点和难点时，在接下来的章节中涉及需要掌握的知识点相对容易，这样做的目的就是增强初学者的学习的信心和提高学习的兴趣。

9.1　基础控件讲解

本节涉及 TextSwitcher 控件、ImageSwitcher 控件，希望读者自行总结它们使用上的异同。

9.1.1　文字交换器

文字交换器（TextSwitcher）控件，实现文本的切换动画。继承自 ViewSwitcher（View 交换器）是交换 TextView 的控件。在代码中如何实现 TextSwitcher 动态切换文本信息的效果呢?首先得实例化一个 TextSwitcher 对象，为 TextSwitcher 指定 ViewSwitc her.ViewFactory 工厂，该工厂会产生转换时需要的 View，为 TextSwitcher 设定显示的内容。在执行该方法的时候，就会切换到下一个 View。当调用 setText()方法时，TextSwitcher 使用设定的动画效果显示新的文字串，从而实现与原先文本内容的切换。常用的方法、事件如表 9-1 所示。

表 9-1　TextSwitcher 常用方法

TextSwitcher 常用方法与事件	功 能 说 明
public TextSwitcher (Context context)	构造函数，创建一个 TextSwitcher 对象 参数：context （应用程序上下文）
public TextSwitcher (Context context, AttributeSet attrs)	构造函数，使用构造函数提供的 context 和 attributes 来创建一个 TextSwitcher 对象 参数：context (应用程序环境)， attrs(属性集合)



（续）

TextSwitcher 常用方法与事件	功 能 说 明
public void addView (View child, int index, ViewGroup.LayoutParams params)	公共方法，根据指定的布局参数新增一个子视图（子项为 TextView） 参数：child（新增的子视图） index（新增子视图的位置） params (新增子视图的布局参数） 抛出异常：IllegalArgumentException （当子视图不是一个 TextView 实例时，则抛出该异常）
public void setCurrentText (CharSequence text)	公共方法，设置当前要显示的文本视图的文字内容，属非动画方式显示 参数：text（需要显示的新文本内容）
public void setText (CharSequence text)	公共方法，设置下一视图的文本内容并切换到下一个视图，可以动态退出当前文本内容并显示下一文本内容 参数：text（需要显示的新文本内容）

下面通过一个示例讲述 TextSwitcher 文本交换器的功能，示例代码运行结果如图 9-1 所示。

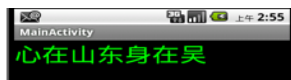


图 9-1　文字交换器示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 9-1 所示。

代码清单 9-1　文字交换器（TextSwitcher）示例（第 9 章\Demo_09_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
      xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      >

   <TextSwitcher
      android:layout_width="match_parent"
      android:layout_height="wrap_content"
      android:id="@+id/textSwitcher1">
       </TextSwitcher>

</LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现文字交换器（TextSwitcher）效果，如代码清单 9-2 所示。

代码清单 9-2　文字交换器（TextSwitcher）示例（第 9 章\Demo_09_01）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.view.View;
import android.view.animation.AnimationUtils;
import android.widget.TextSwitcher;
import android.widget.TextView;
```




```
import android.widget.ViewSwitcher.ViewFactory;
/**
 * TextSwitcher 示例
 * @author
 *
 */
public class MainActivity extends Activity {

	// 当前显示的图片索引
	private int index;
	// 文本数组
	private String[] poemArray = { "心在山东身在吴", "飘蓬江海谩嗟吁",
						"他时若遂凌云志", "敢笑黄巢不丈夫" };

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		// 定义文字切换器
		final TextSwitcher ts = (TextSwitcher) findViewById(R.id.textSwitcher1);

		// 实现并设置工厂内部接口的 makeView 方法，用来显示文本视图
		ts.setFactory(new ViewFactory() {

			@Override
			public View makeView() {
				TextView tv = new TextView(MainActivity.this);
				tv.setTextSize(32);
				tv.setTextColor(Color.GREEN);
				return tv;
			}
		});

		// 设置图片来源
		ts.setText(poemArray[index]);

		// 设置单击监听器
		ts.setOnClickListener(new View.OnClickListener() {

			@Override
			public void onClick(View v) {
				// 单击会切换图片
				index++;
				if (index >= poemArray.length) {
					index = 0;
				}
				ts.setText(poemArray[index]);
			}
		});
```



```
                // 设置切入动画
                ts.setInAnimation(AnimationUtils.loadAnimation(getApplicationContext(),
                        android.R.anim.slide_in_left));

                // 设置切出动画
                ts.setOutAnimation(AnimationUtils.loadAnimation(
                        getApplicationContext(), android.R.anim.slide_out_right));
        }
}
```

9.1.2　图片交换器

图片交换器（ImageSwitcher）是 Android 中控制图片显示效果的控件，可实现幻灯片效果。通常，我们可以在许多网站的首页里看到有个图片循环显示的控件，用来显示站内重点新闻等，在这些网站里大多是采用 JQuery 等 JS 框架提供的循环显示图片的插件，而在 Android 里的 ImageSwitcher 图片交换器提供了类似的功能。ImageSwitcher 涉及的常用方法见表 9-2。

表 9-2　ImageSwitcher 常用方法

ImageSwitcher 方法	功 能 说 明
setImageURI(Uri uri)	设置图片地址
setImageResource(int resid)	设置图片资源库
setImageDrawable(Drawable drawable)	绘制图片

下面通过一个示例讲述 ImageSwitcher 的用法，示例的主要功能就是实现图片幻灯片效果，代码运行结果如图 9-2 所示。



图 9-2　切换图片(ImageSwitcher)示例代码运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 9-3 所示。

代码清单 9-3　切换图片（ImageSwitcher）示例（第 9 章\Demo_09_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
   android:orientation="vertical"
   android:layout_width="fill_parent"
    android:layout_height="fill_parent">

      <ImageSwitcher
        android:id="@+id/imageSwitcher1"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        >
      </ImageSwitcher>

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现切换图片效果，如代码清单 9-4 所示。

代码清单 9-4　切换图片（ImageSwitcher）示例（第 9 章\Demo_09_02）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.WindowManager;
import android.view.animation.AnimationUtils;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.ViewSwitcher.ViewFactory;

/**
 * ImageSwitcher 示例
 *
 * @author
 *
 */
public class MainActivity extends Activity {

    // 当前显示的图片索引
    private int index;
    // 图片数组
    private int[] images = { R.drawable.image1, R.drawable.image2,
            R.drawable.image3, R.drawable.image4, R.drawable.image1 };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        // 全屏设置
        requestWindowFeature(Window.FEATURE_NO_TITLE);
        getWindow().setFlags(WindowManager.LayoutParams.FLAG_FULLSCREEN,
                WindowManager.LayoutParams.FLAG_FULLSCREEN);
        // 获取 main 布局
        setContentView(R.layout.main);
        // 得到 ImageSwitcher 对象
        final ImageSwitcher is = (ImageSwitcher) findViewById(R.id.imageSwitcher1);

        // 实现并设置工厂内部接口的 makeView 方法，用来显示视图
        is.setFactory(new ViewFactory() {
            @Override
            public View makeView() {
                return new ImageView(MainActivity.this);
            }
        });

        // 设置图片来源
```



```
                is.setImageResource(images[index]);

                // 设置单击监听器
                is.setOnClickListener(new View.OnClickListener() {

                    @Override
                    public void onClick(View v) {
                        // 单击会切换图片
                        index++;
                        if (index >= images.length) {
                            index = 0;
                        }
                        is.setImageResource(images[index]);
                    }
                });

                // 设置切入动画
                is.setInAnimation(AnimationUtils.loadAnimation(getApplicationContext(),
                        android.R.anim.slide_in_left));
                // 设置切出动画
                is.setOutAnimation(AnimationUtils.loadAnimation(
                        getApplicationContext(), android.R.anim.slide_out_right));
        }
    }
```

9.2 List详情功能实现

List 详情功能包括“List 详情界面 View 实现”和“List 详情功能 Control 实现”两个层面的内容。

9.2.1 List详情界面View实现

如第 2 章手机订票系统实现效果图（图 2-13）所示，List 详情界面的实现采用 LinearLayout 布局，内嵌 LinearLayout 布局实现界面效果。见代码清单 9-5。

代码清单 9-5 List详情界面View实现（第 9 章\GanaSkyForFlightListDetail）flightlist_detail.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:id="@+id/fligtLisy_items"
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:orientation="vertical"
     android:textSize="15sp"
     android:textColor="#00ff00"
     android:background="@drawable/buttonbgon">

     <!-- 第一行 -->
```




```
<LinearLayout
    android:background="@drawable/buttonbgon"
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <TextView
        android:id="@+id/flightlistDetail_txtTime"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"
        android:layout_centerVertical="true"/>
        <!-- 星期计算 -->
    <TextView
        android:id="@+id/flightlistDetail_txtWeek"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="10dip"
        android:layout_centerVertical="true"/>
    <TextView
        android:id="@+id/flightlistDetail_txtSetCity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="30dip"/>
    <TextView
        android:id="@+id/flightlistDetail_arrow"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"
        android:text="->"   />
    <TextView
        android:id="@+id/flightlistDetail_txtOffCity"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
</LinearLayout>

<LinearLayout
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
    <!-- 第一行航班 -->
    <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dip">
        <TextView
        android:text="航班："
            android:layout_width="wrap_content"
        android:layout_height="wrap_content"
```



```
            android:layout_marginLeft="5dip"/>
             <TextView
              android:id="@+id/flightlistDetail_txtFlightNumber"
                  android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"/>
      </LinearLayout>
       <!-- 第二行 -->
      <LinearLayout
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_marginTop="15dip">
               <TextView
               android:text="机型："
                   android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"/>
             <TextView
               android:id="@+id/flightlistDetail_txtPlaneModel"
                   android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"/>
      </LinearLayout>
       <!-- 第三行 -->
      <LinearLayout
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_marginTop="15dip">
               <TextView
               android:text="起飞："
                   android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"/>
             <TextView
               android:id="@+id/flightlistDetail_txtTakeOffAndFromAirportName"
                   android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"/>
      </LinearLayout>
       <!-- 第四行 -->
      <LinearLayout
              android:orientation="horizontal"
              android:layout_width="fill_parent"
              android:layout_height="wrap_content"
              android:layout_marginTop="15dip">
               <TextView
               android:text="到达："
                   android:layout_width="wrap_content"
```




```
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
         <TextView
           android:id="@+id/flightlistDetail_txtLandingAndToAirportName"
               android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
</LinearLayout>
<!-- 第五行 -->
<LinearLayout
          android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_marginTop="15dip">
           <TextView
           android:text="单价："
               android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
         <TextView
           android:id="@+id/flightlistDetail_txtPrice"
               android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
</LinearLayout>
  <!-- 第六行 -->
  <LinearLayout
          android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_marginTop="15dip">
           <TextView
           android:text="基建/燃油费："
               android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
         <TextView
           android:id="@+id/flightlistDetail_txtAirportConstructionFeeAndFuelSurcharge"
               android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
</LinearLayout>

<!-- 第七行 -->
  <LinearLayout
          android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content"
          android:layout_marginTop="15dip">
           <TextView
```



```
            android:text="是否是电子客票:"
                android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
          <TextView
            android:id="@+id/flightlistDetail_txtElectronicTicket"
                android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
</LinearLayout>

  <!-- 第八行 -->
  <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dip">
          <TextView
            android:text="是否可以改签:"
                android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
          <TextView
            android:id="@+id/flightlistDetail_txtIsChangeTicket"
            android:text="不可以"
                android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:layout_marginLeft="5dip"/>
</LinearLayout>

  <!-- 第九行 -->
  <LinearLayout
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_marginTop="15dip"
        android:gravity="center_vertical">
          <Button
                android:id="@+id/flightlistDetail_btnOrder"
                android:text="预    订"
                android:layout_width="80dip"
                android:layout_height="40dip"
                android:gravity="center"
                android:layout_marginLeft="15dip"/>

          <Button
                android:id="@+id/flightlistDetail_btnReturn"
                android:text="返回查票"
                android:layout_width="80dip"
                android:layout_height="40dip"
```

```
                    android:gravity="center"
                    android:layout_marginLeft="20dip"
                    />
            </LinearLayout>

        </LinearLayout>

    </LinearLayout>
```

9.2.2 List详情功能Control实现

List 信息详情功能 Control 实现，就是航空列表信息的详细显示，实现相对简单，如代码清单 9-6 所示。

代码清单 9-6 List信息详情功能Control实现（第 9 章\GanaSkyForFlightListDetail）FlightListDetail.java（详见本书源代码）

第 10 章　用户信息选择与填写功能实现

由于手机屏幕的限制，我们经常需要通过滚动条滚动信息内容。例如，在手机订票系统中，首先需要让用户选择购买机票的张数，再动态地添加多行文本编辑框。要实现这一效果，可以通过 ScrollView 卷轴视图结合动态添加控件来实现。因此，本章实现的重点就是如何动态添加控件，当添加的子控件超出手机屏幕可视区域，通过 ScrollView 卷轴视图实现滚动效果。

本章涉及的基础知识有：多选按钮（CheckBox）、卷轴视图（ScrollView）的使用。

本章将讲解如下内容。

1）如何动态地为页面添加控件。之前我们的控件都是写在 layout 下的.xml 文件中，属于静态添加的方式，本章将讨论如何在 Activity java 代码中自定义布局并在布局中动态添加子控件的功能。

2）实现多行文本滚动效果。这一知识点将用在填写机票送票地址控件上，若地址较长，动态扩大编辑框的文本容量。

3）如何进行用户身份证信息的验证。前面已使用过正则表达式对注册信息进行验证，本章将继续学习验证，主要是对居民身份证这一特殊且常用的证件的验证。

10.1　基础控件讲解

本节涉及复选框（CheckBox）、卷轴视图（ScrollView）等功能的实现，这些功能对于手机客户端应用程序是至关重要的。

10.1.1　复选框

复选框（CheckBox）允许与用户进行交互，实现多选功能。CheckBox 常用方法如表 10−1 所示。

表 10−1　CheckBox 常用方法

CheckBox 复选框	功 能 说 明
isChecked()	判断复选框是否被选中
setChecked()	设置复选框的 Check 状态
getText()	获取复选框的文本信息
setOnCheckedChangeListener()	设置复选框 Check 状态监听事件
onCheckedChanged()	监听 setOnCheckedChangeListener 事件，需要实现 onCheckedChanged 方法，它主要是取得被选中的复选框的实例，判断复选框的 Check 状态是否改变

CheckBox 复选框示例效果如图 10-1 和图 10-2 所示，示例的主要功能是声明四个 CheckBox 对象以及一个 Button 对象，通过单击 Button 按钮，监听 CheckBox 复选框的 setOnCheckedChangeLisener 事件，实现 onCheckedChanged()方法来获取被选中复选框的文本信息，再将文本信息通过 Toast 消息提示框显示出来。



图 10-1　复选框运行结果—初始化



图 10-2　复选框运行结果—选择子项

首先，介绍如何通过 xml 布局实现多选的效果，如代码清单 10-1 所示。

代码清单 10-1　复选框（CheckBox）示例（第 10 章\Demo_10_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >

     <TextView
          android:id="@+id/TextView1"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="@string/hello"
         />
     <CheckBox
       android:id="@+id/CheckBox1"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="@string/CheckBox1"
     >
     </CheckBox>
     <CheckBox
       android:id="@+id/CheckBox2"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="@string/CheckBox2"
     >
     </CheckBox>
```



```
        <CheckBox
        android:id="@+id/CheckBox3"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/CheckBox3"
        >
        </CheckBox>
        <CheckBox
        android:id="@+id/CheckBox4"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/CheckBox4"
        >
        </CheckBox>
            <Button
              android:id="@+id/button1"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="提交"
            >
            </Button>

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现多选按钮这一效果，如代码清单 10-2 所示。

代码清单 10-2　复选框（CheckBox）示例（第 10 章\Demo_10_01）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.CheckBox;
import android.widget.CompoundButton;
import android.widget.TextView;
import android.widget.Toast;
/**
 * CheckBox 的使用
 *
 * @author
 *
 */
public class MainActivity extends Activity {
        // 用来显示题目
        TextView m_TextView1;
        // 提交按钮
        Button m_Button1;
        // 4 个多选项
```




```
	CheckBox m_CheckBox1;
	CheckBox m_CheckBox2;
	CheckBox m_CheckBox3;
	CheckBox m_CheckBox4;

	/**当 activity 首次创建时响应调用
	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		m_TextView1 = (TextView) findViewById(R.id.TextView1);
		m_Button1 = (Button) findViewById(R.id.button1);

		/* 取得每个 CheckBox 对象 */
		m_CheckBox1 = (CheckBox) findViewById(R.id.CheckBox1);
		m_CheckBox2 = (CheckBox) findViewById(R.id.CheckBox2);
		m_CheckBox3 = (CheckBox) findViewById(R.id.CheckBox3);
		m_CheckBox4 = (CheckBox) findViewById(R.id.CheckBox4);

		// 对每个选项设置事件监听
		m_CheckBox1.setOnCheckedChangeListener(checkBox1);
		m_CheckBox2.setOnCheckedChangeListener(checkBox2);
		m_CheckBox3.setOnCheckedChangeListener(checkBox3);
		m_CheckBox4.setOnCheckedChangeListener(checkBox4);
		m_Button1.setOnClickListener(onClick);

	}

    /**
     * 实现事件监听
     */
	CheckBox.OnCheckedChangeListener checkBox1
								= new CheckBox.OnCheckedChangeListener() {
		@Override
		public void onCheckedChanged(CompoundButton buttonView,
				boolean isChecked) {
			// TODO Auto-generated method stub
			if (m_CheckBox1.isChecked()) {
				DisplayToast("你选择了： " + m_CheckBox1.getText());
			}
		}
	};

	CheckBox.OnCheckedChangeListener checkBox2
								= new CheckBox.OnCheckedChangeListener() {
		@Override
		public void onCheckedChanged(CompoundButton buttonView,
				boolean isChecked) {
			// TODO Auto-generated method stub
```



```
            if (m_CheckBox2.isChecked()) {
                DisplayToast("你选择了： " + m_CheckBox2.getText());
            }
        }
    };

    CheckBox.OnCheckedChangeListener checkBox3
                            = new CheckBox.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked) {
            // TODO Auto-generated method stub
            if (m_CheckBox3.isChecked()) {
                DisplayToast("你选择了： " + m_CheckBox3.getText());
            }
        }
    };

    CheckBox.OnCheckedChangeListener checkBox4
                        = new CheckBox.OnCheckedChangeListener() {
        @Override
        public void onCheckedChanged(CompoundButton buttonView,
                boolean isChecked) {
            // TODO Auto-generated method stub
            if (m_CheckBox4.isChecked()) {
                DisplayToast("你选择了： " + m_CheckBox4.getText());
            }
        }
    };

    Button.OnClickListener onClick = new Button.OnClickListener() {

        @Override
        public void onClick(View arg0) {
            int num = 0;
            String selected="";
            if (m_CheckBox1.isChecked()) {
                num++;
                selected= m_CheckBox1.getText().toString();
            }
            if (m_CheckBox2.isChecked()) {
                num++;
                selected+= ","+m_CheckBox2.getText().toString();
            }
            if (m_CheckBox3.isChecked()) {
                num++;
                selected+= ","+m_CheckBox3.getText().toString();
            }
            if (m_CheckBox4.isChecked()) {
                num++;
```

```
                selected+= ","+m_CheckBox4.getText().toString();
            }

            DisplayToast("您一共选择了" + num + "项。"+"分别是："+selected);
        }

    };

    /* 显示 Toast */
    public void DisplayToast(String str) {
        Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
        // 设置 toast 显示的位置
        toast.setGravity(Gravity.TOP, 0, 220);
        // 显示该 toast
        toast.show();
    }
}
```

10.1.2 卷轴视图

卷轴视图（ScrollView）是一种可供用户滚动的层次结构布局容器，允许显示比实际多的内容。它是 FrameLayout 布局的一种，这意味着它需要在 FrameLayout 上放置具有滚动功能的子元素，子元素可以是一个复杂对象的布局管理器。ScrollView 只支持垂直滚动，通常使用的子元素是垂直方向的 LinearLayout。

ScrollView 卷轴视图的代码实现过程并不复杂，需要声明一个 ScrollView 对象，设置需要滚动内容的布局容器的 xml 属性（如：android:scrollbars="vertical"）。这样，当内容超出屏幕可视范围就会实现滚动效果。下面通过一个示例讲述 ScrollView 的用法，在示例中初始化显示一个 TextView 和一个 Button 控件，单击 Button0 按钮时自动产生多个类似的项，如果屏幕的可视区域不能将内容全部显示，就会通过 ScrollView 实现自动滚动屏幕，显示隐藏子项的效果。示例代码运行结果如图 10-3 所示。



图 10-3　ScrollView 卷轴视图自动添加子项示例运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 10-3 所示。

代码清单 10-3　ScrollView卷轴视图自动添加子项示例（第 10 章\Demo_10_02）main. xml

```
<?xml version="1.0" encoding="utf-8"?>
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/ScrollView"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:scrollbars="vertical">
    <LinearLayout
        android:id="@+id/LinearLayout"
            android:orientation="vertical"
            android:layout_width="fill_parent"
```



```
                android:layout_height="wrap_content">
                <TextView
                    android:id="@+id/TestView"
                    android:layout_width="fill_parent"
                      android:layout_height="wrap_content"
                      android:text="TestView0" />
                <Button android:id="@+id/Button"
                    android:text="Button0"
                     android:layout_width="fill_parent"
                      android:layout_height="wrap_content">
                      </Button>
        </LinearLayout>
</ScrollView>
```

其次，介绍 Activity　java 代码如何配合布局实现 ScrollView 动态添加子项效果，如代码清单 10-4 所示。

代码清单 10-4　ScrollView卷轴视图自动添加子项示例（第 10 章\Demo_10_02）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.os.Handler;
import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.LinearLayout;
import android.widget.ScrollView;
import android.widget.TextView;

/**
 * ScrollView 示例
 * @author
 *
 */
public class MainActivity extends Activity {
        /*初始化对象*/
        private LinearLayout mLayout;
        private ScrollView sView;
        private final Handler mHandler = new Handler();

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);
                //  创建一个线性布局
                mLayout = (LinearLayout) this.findViewById(R.id.LinearLayout);
                //  创建一个 ScrollView 对象
                sView = (ScrollView) this.findViewById(R.id.ScrollView);
```

```
            Button mBtn = (Button) this.findViewById(R.id.Button);
            // 添加单击事件监听
            mBtn.setOnClickListener(mClickListener);
    }
/**
 *  键盘响应事件
 */
    public boolean onKeyDown(int keyCode, KeyEvent event){
            Button b = (Button) this.getCurrentFocus();
            int count = mLayout.getChildCount();
            Button bm = (Button) mLayout.getChildAt(count-1);
        //当按下上方向键且是 mBtn 按钮
            if(keyCode==KeyEvent.KEYCODE_DPAD_UP && b.getId()==R.id.Button){
                bm.requestFocus();
                return true;
            }else if(keyCode==KeyEvent.KEYCODE_DPAD_DOWN
                    && b.getId()==bm.getId()){
                this.findViewById(R.id.Button).requestFocus();
                return true;
            }
            return false;
    }
    // Button 事件监听，当单击第一个按钮时增加一个 Button 和一个 TextView
     Button.OnClickListener mClickListener = new Button.OnClickListener() {

            private int index = 1;
            @Override
            public void onClick(View v) {
                TextView tView = new TextView(MainActivity.this);//定义一个 TextView
                tView.setText("TextView" + index);//设置 TextView 的文本信息
                //设置线性布局的属性
                LinearLayout.LayoutParams params = new LinearLayout.LayoutParams(
                        LinearLayout.LayoutParams.FILL_PARENT,
                        LinearLayout.LayoutParams.WRAP_CONTENT);
                mLayout.addView(tView, params);//添加一个 TextView 控件
                Button button = new Button(MainActivity.this);//定义一个 Button
                button.setText("Button" + index);//设置 Button 的文本信息
                button.setId(index++);
                mLayout.addView(button, params);//添加一个 Button 控件
                mHandler.post(mScrollToButton);//传递一个消息进行滚动
            }

    };
    //启动进程
    private Runnable mScrollToButton = new Runnable() {
            @Override
            public void run() {
                int off = mLayout.getMeasuredHeight() - sView.getHeight();
                if (off > 0) {
                    sView.scrollTo(0, off);//改变滚动条的位置
```



```
                    }
                }

            };

        }
```

10.2　重点剖析

本节介绍需要重点掌握的 Android 编程方法与编程思路，主要介绍如何动态增加控件、多行文本滚动、用户身份证验证等子功能的实现。

10.2.1　如何动态增加控件

动态地为布局添加子控件，可以通过捕获当前的布局，动态地在布局中添加控件来实现这一功能，增加的子控件数量可由用户自定义。实现动态添加控件的重点是：为界面的哪一层布局增加子项，子项如何排列等问题。在实现过程中，通常需要捕获到要增加子项控件的布局 ID，然后把需要添加的子项控件通过调用 addView()方法自动添加进去。下面通过一个示例，讲述具体实现过程，示例代码运行结果如图 10-4 和图 10-5 所示。



图 10-4　如何动态添加控件示例——初始化



图 10-5　如何动态添加控件示例——单击按钮动态添加

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 10-5 所示。

代码清单 10-5　如何动态添加控件示例（第 10 章\Demo_10_03）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:orientation="vertical"
     android:textSize="15sp">
        <LinearLayout
         android:orientation="horizontal"
         android:layout_width="fill_parent"
         android:layout_height="wrap_content">
            <TextView
                android:text="增加几行文本框："
```




```
                android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="1dip"/>
            <EditText
                  android:id="@+id/ed_Number"
                   android:layout_width="60dip"
                   android:layout_height="wrap_content"
                   android:focusable="true"
                   android:textSize="11sp"/>
                   <Button
                android:id="@+id/btn_Add"
                 android:text="确 定"
                 android:layout_width="80dip"
                 android:layout_height="40dip"
                 android:gravity="center"
                 android:layout_marginLeft="15dip"/>
        </LinearLayout>
         <LinearLayout
                android:id="@+id/lineLayOut"
                  android:orientation="vertical"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginLeft="5dip">
       <!-- 第一行 -->
                <LinearLayout
                            android:orientation="horizontal"
                            android:layout_width="fill_parent"
                            android:layout_height="wrap_content" >
                         <EditText
                               android:id="@+id/ed_add1"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:textSize="11sp"
                                android:focusable="true"
                                android:hint="第 1 行第 1 列"/>

                         <EditText
                               android:id="@+id/ed_add2"
                                android:layout_width="wrap_content"
                                android:layout_height="wrap_content"
                                android:hint="第 1 行第 2 列"
                                android:focusable="true"
                                android:textSize="11sp"/>

                   </LinearLayout>
          </LinearLayout>

    </LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现动态添加控件这一功能，如代码清单 10-6 所示。



代码清单 10-6　如何动态添加控件示例（第 10 章\Demo_10_03）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.ViewGroup;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
/**
 * 添加子项示例
 * @author
 *
 */
public class MainActivity extends Activity {
	private EditText ed_Number;
	private Button btn_Add;
	private LinearLayout linearLayoutTotal;

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);
		/* 获取对象 */
		ed_Number = (EditText) findViewById(R.id.ed_Number);
		btn_Add = (Button) findViewById(R.id.btn_Add);
		linearLayoutTotal = (LinearLayout) findViewById(R.id.lineLayOut);

		/* 设置监听事件 */
		btn_Add.setOnClickListener(Add);
	}
	/**
	 * 实现监听
	 */
	Button.OnClickListener Add = new Button.OnClickListener() {

		@Override
		public void onClick(View arg0) {
			// TODO Auto-generated method stub
		  //增加子项
			AddItem(Integer.parseInt(ed_Number.getText().toString()));
		}
	};

	/**
	 * 添加子项
	 *
	 * @param num
```

```
         */
        private void AddItem(int num) {
            if (num != 0) {
                final int WRAP_CONTENT = ViewGroup.LayoutParams.WRAP_CONTENT;
                final int FILL_PARENT = ViewGroup.LayoutParams.FILL_PARENT;
                linearLayoutTotal.removeAllViews();
                for (int i = 0; i < num; i++) {
                    LinearLayout linearLayout1 = new LinearLayout(this);
                    linearLayout1.setOrientation(LinearLayout.HORIZONTAL);
                    linearLayoutTotal
                            .addView(linearLayout1, new LinearLayout.LayoutParams(
                                    FILL_PARENT, WRAP_CONTENT));
                 //动态增加文本框（可以换成其他子项）
                    EditText editTextName = new EditText(this);
                    editTextName.setHint("第" + (i + 1) + "行第" + 1+ "列");
                    editTextName.setTextSize(11);
                    linearLayout1.addView(editTextName,
                            new LinearLayout.LayoutParams(WRAP_CONTENT,
                                    WRAP_CONTENT));

                    EditText editTextCardId = new EditText(this);
                    editTextCardId.setTextSize(11);
                    editTextCardId.setHint("第" + (i + 1) + "行第" + 2+ "列");
                    linearLayout1.addView(editTextCardId,
                            new LinearLayout.LayoutParams(WRAP_CONTENT,
                                    WRAP_CONTENT));
                }
            }
        }

    }
```

10.2.2　多行文本滚动实现

在填写联系地址的时候，由于信息量较大，单行难以显示详细的送票地址。且由于手机屏幕大小的限制，需要对 EditText 文本编辑框进行相应的处理，实现多行文本滚动效果，来减少手机屏幕空间位置的占用。如何实现多行文本编辑框滚动效果，相信通过上面章节的讲解，大家会想到通过 ScrollView+EditText 来实现。

下面通过一个示例讲述多行文本滚动效果的实现过程，示例的主要功能是使用 ScrollView 实现 EditText 编辑框文本信息的滚动，这需要设置文本编辑框（EditText）可多行显示文本，还需要设置从第几行开始实现文本滚动效果，示例代码运行结果如图 10-6 所示。

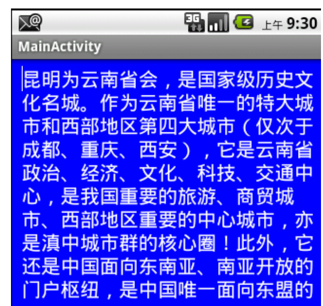


图 10-6　多行文本滚动示例运行结果

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 10-7 所示。

代码清单 10-7　身份证验证示例（第 10 章\Demo_10_04）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
```



```
<ScrollView
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:id="@+id/ScrollView"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     >
     <LinearLayout
          android:id="@+id/LinearLayout"
           android:orientation="vertical"
           android:layout_width="fill_parent"
           android:layout_height="wrap_content">
           <EditText
                android:id="@+id/textview1"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:scrollbars="vertical"
                android:singleLine="false"
                android:maxLines="10"
                android:textColor="#FF0000"
          />
     </LinearLayout>
</ScrollView>
```

其次，介绍 Activity java 代码如何配合布局实现多行文本滚动效果，如代码清单 10-8 所示。

代码清单 10-8　多行文本滚动示例（第 10 章\Demo_10_04）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.text.method.ScrollingMovementMethod;
import android.widget.TextView;

/**
 * Demo_10_04 多行文本滚动实现
 *
 * @author
 */
public class MainActivity extends Activity {
     /* 声明 TextView 对象 */
     private TextView textview1;

     /** 当 activity 首次创建时响应调用
     @Override
     public void onCreate(Bundle savedInstanceState) {
          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          /* 获得 TextView 对象 */
```

```
                textview1 = (TextView) this.findViewById(R.id.textview1);

                String word = "昆明为云南省会，是国家级历史文化名城。"
                        + "作为云南省唯一的特大城市和西部地区第四大城市（仅次于成都、重庆、西
                        安）。"
                        + "它是云南省政治、经济、文化、科技、交通中心，是我国重要的旅游、商贸
                        城市、"
                        + "西部地区重要的中心城市，亦是滇中城市群的核心圈！"
                        + "此外，它还是中国面向东南亚、南亚开放的门户枢纽，是中国唯一面向东盟
                        的大都市。"
                        + "因夏无酷暑、冬无严寒、气候宜人，具有典型的温带气候特点，"
                        +"城区温度在 0～29℃之间，年温差为全国最小，"
                        + "这样的全球少有的气候特征使昆明以“春城”而享誉中外";

                /* 设置文本的颜色 */
                textview1.setTextColor(Color.WHITE);
                /* 设置文字大小 */
                textview1.setTextSize(20);
                /* 设置文字背景 */
                textview1.setBackgroundColor(Color.BLUE);
                /* 设置 TextView 显示的文字 */
                textview1.setText(word);
                textview1.setMovementMethod(ScrollingMovementMethod.getInstance());

        }
}
```

10.2.3 用户身份证的验证

用户身份证验证，可以使用正则表达式，也可以使用纯 java 代码方式实现。本示例主要使用纯 java 方式实现身份证的验证，读者可以尝试使用前面讲过的正则表达式实现身份证的验证，示例代码运行结果如图 10-7、10-8 所示。



图 10-7 身份证验证示例代码运行结果（1）——输入错误的身份证号



图 10-8 身份证验证示例代码运行结果（2）———输入正确的身份证号

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 10-9 所示。

代码清单 10-9 身份证验证示例（第 10 章\Demo_10_05）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
```



```
xmlns:android="http://schemas.android.com/apk/res/android"
 android:layout_width="fill_parent"
 android:layout_height="fill_parent"
 android:orientation="vertical"
 android:textSize="15sp">

        <TextView
            android:text="输入您的身份证："
                android:layout_width="wrap_content"
          android:layout_height="wrap_content"
          android:layout_marginLeft="1dip"/>
          <EditText
                  android:id="@+id/ed_CardID"
                   android:layout_width="fill_parent"
                   android:layout_height="wrap_content"
                   android:focusable="true"
                   android:textSize="11sp"/>
                   <Button
              android:id="@+id/btn_Add"
               android:text="确  定"
               android:layout_width="80dip"
               android:layout_height="40dip"
               android:gravity="center"
               android:layout_marginLeft="15dip"/>
</LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现用户身份证验证这一功能，如代码清单 10-10 所示。

代码清单 10-10　身份证验证示例（第 10 章\Demo_10_05）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
/**
 *  身份证验证
 * @author
 *
 */
public class MainActivity extends Activity {
    private EditText ed_CardID;
    private Button btn_Add;

    VerifyIdCard verifyIdCard = new VerifyIdCard();
```




```
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main);

                /* 获取对象 */
                ed_CardID = (EditText) findViewById(R.id.ed_CardID);
                  btn_Add = (Button) findViewById(R.id.btn_Add);
                  /* 设置监听事件 */
                  btn_Add.setOnClickListener(Add);
        }

        /**
         * 实现监听
         */
         Button.OnClickListener Add = new Button.OnClickListener() {

                  @Override
                  public void onClick(View arg0) {

                          //验证
                          if(verifyIdCard.verify(ed_CardID.getText().toString().toUpperCase() )){
                                  DisplayToast("您输入正确了！ ");
                          }else{
                                  DisplayToast("您输入错误了！ ");
                          }

                  }
         };

         /* 显示 Toast */
         public void DisplayToast(String str) {
                  Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
                  // 设置 toast 显示的位置
                  toast.setGravity(Gravity.TOP, 0, 220);
                  // 显示该 toast
                  toast.show();
         }
}
```

代码清单 10-11　身份证验证示例（第 10 章\Demo_10_05）　VerifyIdCard.java

```
package com.chen.android;
/**
 * 身份证验证
 * @author
 *
 */
public class VerifyIdCard {
```



```
// wi =2(n-1)(mod 11);加权因子
final int[] wi = { 7, 9, 10, 5, 8, 4, 2, 1, 6, 3, 7, 9, 10, 5, 8, 4, 2, 1 };
// 校验码
final int[] vi = { 1, 0, 'X', 9, 8, 7, 6, 5, 4, 3, 2 };
private int[] ai = new int[18];

public VerifyIdCard() {
}

/**
 * 15、18 位的验证
 * @param idcard
 * @return
 */
public boolean verify(String idcard) {
        if (idcard.length() == 15) {
                idcard = uptoeighteen(idcard);
        }
        if (idcard.length() != 18) {
                return false;
        }
        String verify = idcard.substring(17, 18);
        if (verify.equals(getVerify(idcard))) {
                return true;
        }
        return false;
}

// 15 位转 18 位
public String uptoeighteen(String fifteen) {
        StringBuffer eighteen = new StringBuffer(fifteen);
        eighteen = eighteen.insert(6, "19");
        return eighteen.toString();
}

// 计算最后一位校验值
public String getVerify(String eighteen) {
        int remain = 0;
        if (eighteen.length() == 18) {
                eighteen = eighteen.substring(0, 17);
        }
        if (eighteen.length() == 17) {
                int sum = 0;
                for (int i = 0; i < 17; i++) {
                        String k = eighteen.substring(i, i + 1);
                        ai[i] = Integer.valueOf(k);
                }
                for (int i = 0; i < 17; i++) {
                        sum += wi[i] * ai[i];
                }
```

```
                remain = sum % 11;
            }
            return remain == 2 ? "X" : String.valueOf(vi[remain]);

        }
    }
```

10.3　用户信息选择与填写（机票预订）实现

机票预订包括 View、Model、功能 Control、后台 ASP 等环节，本节给出了一个完整的实现过程。

10.3.1　机票预订View实现

如第 2 章手机订票系统实现效果图（如图 2-14）所示，机票预订功能的 View 实现采用 Linear Layout 布局，内嵌两层 LinearLayout，可动态添加子项，子项超出可视区域，实现滚动效果。详见代码清单 10-12。

代码清单 10-12　预订功能View实现（第 10 章\GanaSkyForTicketOrder）orderticket.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android "
      android:layout_width="fill_parent"
      android:layout_height="fill_parent"
      android:orientation="vertical"
      android:textSize="15sp"
      android:background="@drawable/buttonbgon">
      <!-- 第一行 -->
      <LinearLayout
           android:background="@drawable/buttonbgon"
          android:orientation="horizontal"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content">
          <TextView
                android:id="@+id/orderticket_txtTime"
                android:layout_width="wrap_content"
               android:layout_height="wrap_content"
                 android:layout_marginLeft="5dip"
                 android:layout_centerVertical="true"
                 android:textColor="#ffffff "
                 android:textSize="15sp"/>
                 <!-- 星期计算 -->
              <TextView
                android:id="@+id/orderticket_txtWeek"
                android:layout_width="wrap_content"
               android:layout_height="wrap_content"
                 android:layout_marginLeft="10dip"
                 android:layout_centerVertical="true"
```



```
            android:textColor="#ffffff "
            android:textSize="15sp"/>
        <TextView
            android:id="@+id/orderticket_txtSetCity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="30dip"/>
        <TextView
            android:id="@+id/orderticket_arrow"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"
            android:text="->"   />
        <TextView
            android:id="@+id/orderticket_txtOffCity"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="5dip"
            android:textColor="#ffffff"
            android:textSize="15sp"/>
    </LinearLayout>
    <LinearLayout
        android:background="@drawable/buttonbgon"
        android:orientation="horizontal"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content">
        <TextView
            android:text="证件类型:"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:layout_marginLeft="1dip"
            android:textColor="#ffffff "
            android:textSize="15sp"/>
        <Spinner
            android:id="@+id/orderticket_selectCardType"
            android:gravity="right"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content" />
        <TextView
            android:text="订票人数： "
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:gravity="right"
            android:layout_marginLeft="1dip"
            android:textColor="#ffffff "
            android:textSize="15sp"/>
        <Spinner
            android:id="@+id/orderticket_choicePersonNumber"
            android:gravity="right"
            android:layout_width="69dip"
```

```
                    android:layout_height="wrap_content"
                    android:layout_marginRight="1dip"/>
        </LinearLayout>
      <ScrollView
           xmlns:android="http://schemas.android.com/apk/res/android "
            android:id="@+id/ScrollView"
            android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            >
            <LinearLayout
                android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:scrollbars="vertical">
             <LinearLayout
                android:orientation="vertical"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:id="@+id/orderticket_lineLayOut"
                android:layout_marginLeft="5dip">
                <!-- 第一行 -->
                <LinearLayout
                        android:orientation="horizontal"
                        android:layout_width="fill_parent"
                        android:layout_height="wrap_content" >
                        <EditText
                              android:id="@+id/orderticket_userName1"
                               android:layout_width="wrap_content"
                               android:layout_height="wrap_content"
                               android:textSize="11sp"
                               android:focusable="true"
                               android:hint="请输入身份证上对应的姓名"/>

                         <EditText
                              android:id="@+id/orderticket_userCardId1"
                               android:layout_width="wrap_content"
                               android:layout_height="wrap_content"
                               android:hint="请输入正确的身份证号码"
                               android:focusable="true"
                               android:textSize="11sp"/>

                  </LinearLayout>
             </LinearLayout>
             <LinearLayout
                 android:background="@drawable/buttonbgon"
                android:orientation="horizontal"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content">
                 <CheckBox
                    android:id="@+id/orderticket_IsSentTicket"
```



```
            android:text="是否送票"
            android:checked="true"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content"
            android:textColor="#ffffff "
             android:textSize="15sp" />
          <!-- 地址可多行显示且可滚动 -->
         <ScrollView
               xmlns:android="http://schemas.android.com/apk/res/android "
                android:id="@+id/ScrollViewofAddress"
                android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                >
                <LinearLayout
                      android:id="@+id/LinearLayout"
                       android:orientation="vertical"
                       android:layout_width="fill_parent"
                       android:layout_height="wrap_content">
                      <EditText
                          android:id="@+id/orderticket_edAddress"
                          android:hint="送票地址"
                          android:layout_width="fill_parent"
                          android:layout_height="wrap_content"
                          android:layout_marginLeft="10dip"
                          android:scrollbars="vertical"
                           android:singleLine="false"
                        android:maxLines="2"
                          />
           </LinearLayout>
    </ScrollView>
    </LinearLayout>
   <LinearLayout
        android:background="@drawable/buttonbgon"
       android:orientation="horizontal"
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:layout_marginTop="10dip">
       <Button
           android:id="@+id/orderticket_btnOK"
            android:text="确  定"
            android:layout_width="80dip"
            android:layout_height="45dip"
            android:gravity="center"
            android:layout_marginLeft="15dip"
            android:textColor="#ffffff "
             android:textSize="15sp"/>
        <Button
            android:id="@+id/orderticket_btnReturn"
            android:text="返回查票"
            android:layout_width="80dip"
```

```
                        android:layout_height="45dip"
                        android:gravity="center"
                        android:layout_marginLeft="20dip"
                        android:textColor="#ffffff "
                         android:textSize="15sp"/>

            </LinearLayout>
        </LinearLayout>
    </ScrollView>
</LinearLayout>
```

10.3.2 机票预订Model实现

机票预订涉及乘客信息，因此有必要建一个专门管理乘客信息的类，如代码清单10-13所示。

代码清单 10-13 机票预订Model实现（第 10 章\GanaSkyForTicketOrder）PassengerInfo.java

```
package com.ganasky.model;
/*
 *乘客信息类
 */
public class PassengerInfo {
    //乘客姓名
    private String passengerName;
    //乘客 ID
    private String passengerCardID;

    public String getPassengerName() {
        return passengerName;
    }
    public void setPassengerName(String passengerName) {
        this.passengerName = passengerName;
    }
    public String getPassengerCardID() {
        return passengerCardID;
    }
    public void setPassengerCardID(String passengerCardID) {
        this.passengerCardID = passengerCardID;
    }
}
```

10.3.3 机票预订功能Control实现

机票预订功能的 Control 实现，就是机票预订功能的 Activity .java 代码实现，如代码清单 10-14、10-15 所示。在实现过程中需要注意以下几点：

1）如何根据用户选择的用户人数，动态添加编辑文本框以方便用户输入信息，这一点在基础知识讲解和重点知识讲解部分已涉及；在单击预订按钮的时候，如何获取动态添加的编辑框的用户输入信息，也是实现的关键。



2）预订机票功能只有已注册且登录了该平台后的用户才能进行操作，因此需要判断用户是否已经登录，若未登录切换到登录界面，若已登录，则需要获取用户信息并传入到该页面中。若未注册，注册后方可预订机票。

3）机票预订界面要接收来自三个页面（用户登录、用户注册、机票详情界面）的信息，因此该页面值的传入与信息的组装是实现的难点。

代码清单 10-14　机票预订功能Control实现（第 10 章\GanaSkyForTicketOrder）OrderTicket. Java（详见本书源代码）

代码清单 10-15　机票预订功能Control实现（第 10 章\GanaSkyForTicketOrder）VerifyIdCard.java（详见本书源代码）

10.3.4　机票预订后台ASP实现

机票预订功能的 ASP 实现，就是把乘客选择的航班信息、舱位信息及乘客身份证信息提交到后台处理，实现这一过程需要调用底层 Eterm 指令，生成 Pnr 的过程。读者不必太深究怎么实现机票底层功能，可以使用第三方接口。如果想进一步学习机票预订的底层实现，可以参考相关方面的资料。具体实现见代码清单 10-16。

代码清单 10-16　机票预订功能ASP实现（第 10 章\GanaSkyForTicketOrder）OrderTicket.aspx.cs（详见本书源中的代码）

机票预订功能执行 ASP 后台代码后返回的 XML 格式的数据分两种。一种是生成 Pnr，预订成功，显示如代码清单 10-17 所示。另一种是中航信系统容易变动，无法预订，则显示代码清单 10-18 所示。

代码清单 10-17　机票预订功能执行ASP后台代码后返回的XML格式的数据（第 10 章\GanaSkyForTicketOrder\assets\data）——预订成功

```
<?xml version="1.0" encoding="utf-8"?>
<Result>
    等待出票,我们将尽快与您联系。
    您要支付的机票金额是 500  元。
      若要其他服务如改签，请拨打电话：0871-5935000
</Result>
```

代码清单 10-18　机票预订功能执行ASP后台代码后返回的XML格式的数据　（第 10 章\ GanaSkyForTicketOrder\assets\data）——预订失败

```
<?xml version="1.0" encoding="utf-8"?>
<Result>
未能订票请原谅。请稍候再试
</Result>
```

第 11 章　订购成功实现

细心的读者会发现，在购票人提交个人信息后，应该是支付环节。但由于目前手机支付还不够完善，电子支付存在安全性问题，通过手机进行大宗商品支付的用户相对较少。很多商家都选用送货上门，用户再付款的方式，本项目是学习型项目，因此也没有涉及支付环节，本项目采用货到付款的方式。

在本章的基础控件讲解部分，将重点讲解状态栏提示（Notification、NotificationManager）功能，它是一个很不错的提示工具。基础控件部分还讲解了拖动条（SeekBar）和图片长廊（Gallery），这两个控件在项目中未涉及（Gallery 在第 3 章数据适配器的示例中曾使用过），但为了让本书控件讲解部分全面完整，在这里也一并讲解。

11.1　基础控件讲解

本节涉及状态栏提示（Notification、NotificationManager）工具、拖动条（SeekBar）控件、图片长廊（Gallery）组件，这些组件在其他项目中也会经常使用。

11.1.1　状态栏提示

用过 Android 操作系统的用户，对状态栏消息提示功能并不陌生，我们可以在项目中增加状态栏消息提示。状态栏提示（Notification、NotificationManager）是一个不错的提示工具，不干扰正常的操作，在状态栏提示过后还可以单击链接查看更详细的内容。Notification 状态栏提示功能的主要组成部分如下。

1）Icon：状态栏提示的图标。

2）Ticker Text：首次加载 Notification 的时候，在状态栏上滚动的字幕如果很长，会自动分割滚动。

3）Content Title：Notification 展开后的标题。

4）Content Text：Notification 展开后的内容。

下面通过一个示例，讲述状态栏提示的用法，示例代码运行结果如图 11−1 和图 11−2 所示。



图 11−1　状态栏消息提示示例（1）——单击按钮弹出通知





图 11-2　状态栏消息提示示例（2）——单击信息切换页面

首先，介绍如何通过 xml 布局实现状态栏提示功能，如代码清单 11-1、11-2 所示。

代码清单 11-1　状态栏（Notification）消息提示示例（第 11 章\Demo_11_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
    <TextView
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="@string/hello" />
     <Button android:text="Button01"
        android:id="@+id/Button01"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content">
     </Button>

</LinearLayout>
```

代码清单 11-2　状态栏（Notification）消息提示示例（第 11 章\Demo_11_01）main2.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
  xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <TextView
        android:id="@+id/tv_result"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:text="谢谢使用手机订票系统" />
</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 Notification 状态栏信息提示效果，如代码清单 11-3、11-4 所示。

代码清单 11-3　状态栏（Notification）消息提示示例（第 11 章\Demo_11_01）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.app.Notification;
import android.app.NotificationManager;
```

```
import android.app.PendingIntent;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
/**
 *   状态栏提示（Notification、NotificationManager）示例
 * @author
 *
 */
public class MainActivity extends Activity {
    /*定义变量*/
    private Button btn1;
    private Notification notification;
    private NotificationManager notificationManager;
    private Intent intent;
    private PendingIntent pendIntent;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        btn1 = (Button) this.findViewById(R.id.Button01);

        // 获取系统服务（消息管理）
        notificationManager = (NotificationManager) this
                .getSystemService(NOTIFICATION_SERVICE);
        // 单击通知时转移内容
        intent = new Intent(this, Activity2.class);

        // 设置传递的参数
        intent.putExtra("result", "您好！谢谢使用本系统");

        // 设置单击通知时显示内容的类
        pendIntent = PendingIntent.getActivity(this, 0, intent, 0);
        notification = new Notification();
        btn1.setOnClickListener(new OnClickListener() {

            @Override
            public void onClick(View v) {

                // 设置在状态栏显示的图标
                notification.icon = R.drawable.icon;
                //设置在状态栏显示的内容
                notification.tickerText = "Button1 通知内容.......";
                // 默认的声音
                notification.defaults = Notification.DEFAULT_SOUND;
                // 设置通知显示的参数
                notification.setLatestEventInfo(MainActivity.this,
                        "Button1", "Button1 通知", pendIntent);
                // 执行通知
                notificationManager.notify(0, notification);
```



```
                    }
                });
            }
        }
```

代码清单 11-4　状态栏（Notification）消息提示示例（第 11 章\Demo_11_01）Activity2.java

```
package com.chen.android;

import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.widget.TextView;
/**
 *  点开状态栏的通知跳转到该页面
 * @author
 *
 */
public class Activity2 extends Activity {
        String result;
        TextView tv_result;
        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                setContentView(R.layout.main2);

                 /*接收参数*/
                Intent intent = getIntent();
                result = intent.getStringExtra("result");

                 /*显示信息*/
                tv_result=(TextView)this.findViewById(R.id.tv_result);
                tv_result.setText(result);
        }
}
```

11.1.2　拖动条

拖动条（SeekBar）是 ProgressBar 的一个子类，分不确定（indeterminate=true）和确定（indeterminate=false）拖动条两种。默认值是确定拖动条的方式，确定的拖动条可以设置进度值，其他方式设置进度值无效。SeekBar 常用属性如表 11-1 所示。

表 11-1　SeekBar 常用属性

SeekBar 属性	功 能 说 明
style	显示方式，取值： ?android:attr/progressBarStyleLarge / progressBarStyle / progressBarStyleSmall / progressBarStyleHorizontal
android:max=100	设置进度条最大值
android:progress=50	设置进度条主进度当前值
android:secondaryProgress=75	设置进度条次进度当前值
android:thumb	拇指跟随图标
android:thumbOffset	设置允许的轨道范围扩展到拇指的偏移量

通过设置 setOnSeekBarChangeListener(SeekBar.OnSeekBarChangeListener) 监听事件，可监听开始拖拽、停止拖拽以及拖拽中进度条的值是否是用户改变的值等参数。SetOnSeekBarChangeListener 中需要实现的方法见表 11-2。

表 11-2　SeekBar 中的 setOnSeekBarChangeListener 事件常用方法

setOnSeekBarChangeListener 中需要实现的方法	功 能 说 明
onProgressChanged(SeekBar seekBar，int progress，　boolean fromUser)	拖拽中
onStartTrackingTouch(SeekBar seekBar)	开始拖拽
onStopTrackingTouch(SeekBar seekBar)	拖拽完成

下面通过一个示例，讲述拖动条（SeekBar）的用法，示例代码运行结果如图 11-3 所示。

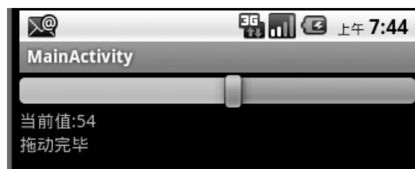


图 11-3　拖动条（SeekBar）示例代码运行结果

首先，介绍如何通过 xml 布局实现拖动条效果，如代码清单 11-5 所示。

代码清单 11-5　拖动条（SeekBar）示例（第 11 章\Demo_11_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
      android:orientation="vertical"
      android:layout_width="fill_parent"
      android:layout_height="fill_parent">
      <SeekBar android:id="@+id/SeekBar01"
          android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:max="100"
            android:progress="50"
            android:secondaryProgress="100">
       </SeekBar>
      <TextView
          android:id="@+id/TextView1"
           android:layout_width="fill_parent"
            android:layout_height="wrap_content"
            android:text="" />
       <TextView android:id="@+id/TextView2"
            android:layout_width="fill_parent"
             android:layout_height="wrap_content"
              android:text="" />
</LinearLayout>
```



其次，介绍 Activity　java 代码如何配合布局实现拖动条效果，如代码清单 11-6 所示。

代码清单 11-6　拖动条（SeekBar）示例（第 11 章\Demo_11_02）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.widget.SeekBar;
import android.widget.TextView;
/**
 * SeekBar 示例
 * 继承 Activity
 * 实现 OnSeekBarChangeListener 接口
 */
public class MainActivity extends Activity
                implements SeekBar.OnSeekBarChangeListener{
    /** 初始化控件对象 */
     private SeekBar seekBar;
     private TextView textView1,textView2;
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);

        /** 获取布局界面控件 */
        seekBar = (SeekBar) this.findViewById(R.id.SeekBar01);
        textView1 = (TextView) this.findViewById(R.id.TextView1);
        textView2 = (TextView) this.findViewById(R.id.TextView2);

      //添加事件监听
        seekBar.setOnSeekBarChangeListener(this);
    }
    //拖动中
     @Override
     public void onProgressChanged(SeekBar seekBar, int progress,
                 boolean fromUser) {
           this.textView1.setText("当前值:"+progress);

     }
     //开始拖动
     @Override
     public void onStartTrackingTouch(SeekBar seekBar) {
           this.textView2.setText("拖动中...");

     }
     //结束拖动
     @Override
     public void onStopTrackingTouch(SeekBar seekBar) {
           this.textView2.setText("拖动完毕");

     }
}
```

11.1.3 循环播放图片列表效果

循环播放图片列表效果（Gallery）组件主要用于横向显示图像列表，实现图片的浏览功能。Gallery 组件在第 3 章数据适配器的使用示例中曾涉及，在这里将更为系统地学习 Gallery 组件的用法。

在第 3 章数据适配器 Gallery 的使用示例中，运行效果总感觉有些生硬，这一节将进一步改进示例，讲述 ImageSwitcher+Gallery 的用法，实现幻灯片循环显示图片，单击某一个 Gallery 组件中的图像时，在上方放大选择的图片的效果。当循环滚动到最后一张图片时，可从第一张图片开始再重新显示图片。示例代码运行结果如图 11-4 和图 11-5 所示。



图 11-4　ImageSwitcher+Gallery 示例——初始化界面



图 11-5　ImageSwitcher+Gallery 示例——单击某一图片，动态放大

首先，介绍如何通过 xml 布局实现循环显示图片效果，如代码清单 11-7、11-8 所示。

代码清单 11-7　ImageSwitcher+Gallery使用示例（第 11 章\Demo_11_03）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<Gallery
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:id="@+id/Gallery"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">
</Gallery>
```

代码清单 11-8　ImageSwitcher+Gallery使用示例（第 11 章\Demo_11_03）image_show.xml

```
<?xml version="1.0" encoding="utf-8"?>
<RelativeLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent">

    <ImageSwitcher android:id="@+id/ImageSwitcher01"
        android:layout_height="fill_parent"
        android:layout_width="fill_parent"
        android:layout_alignParentTop="true"
        android:layout_alignParentLeft="true">
```



```
        </ImageSwitcher>

        <Gallery
          android:id="@+id/gallery"
          android:background="#55000000"
          android:layout_width="fill_parent"
          android:layout_height="60dp"
          android:layout_alignParentBottom="true"
          android:layout_alignParentLeft="true"
          android:gravity="center_vertical"
          android:spacing="16dp" />

</RelativeLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现拖动效果，如代码清单 11-9 至代码清单 11-11 所示。

代码清单 11-9　ImageSwitcher+Gallery使用示例（第 11 章\Demo_11_03）MainActivity.java

```
package com.chen.android;

import android.widget.*;
import android.app.Activity;
import android.content.Intent;
import android.os.Bundle;
import android.view.View;
import android.view.View.OnClickListener;
/**
 *   ImageSwitcher+Gallery 使用示例
 * @author
 *
 */
public class MainActivity extends Activity {

    OnClickListener listener0 = null;
    Button button0;

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        // 单击按钮跳转到 ImageShowActivity
        listener0 = new OnClickListener() {
            public void onClick(View v) {
                Intent intent = new Intent(MainActivity.this,
                        ImageShowActivity.class);
                startActivity(intent);
            }
        };

        button0 = (Button) findViewById(R.id.image_show_button);
        button0.setOnClickListener(listener0);
    }
}
```

代码清单 11-10　ImageSwitcher+Gallery 使用示例（第 11 章\Demo_11_03）ImageShowActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.view.Window;
import android.view.animation.AnimationUtils;
import android.widget.AdapterView;
import android.widget.Gallery;
import android.widget.ImageSwitcher;
import android.widget.ImageView;
import android.widget.AdapterView.OnItemSelectedListener;
import android.widget.RelativeLayout.LayoutParams;
import android.widget.ViewSwitcher.ViewFactory;

/**
 * ImageSwitcher+ Gallery 实现查看相册画面
 * 继承 Activity
 *实现 ViewFactory 和 OnItemSelectedListener 接口
 */
public class ImageShowActivity extends Activity implements ViewFactory,
        OnItemSelectedListener {
    /** 当 aetivity 首次创建时响应调用*/
    ImageSwitcher mSwitcher;
    //ImageSwitcher 数据源
    private Integer[] mImageIds = { R.drawable.f1,
            R.drawable.f2, R.drawable.f3, R.drawable.f4 };

    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        requestWindowFeature(Window.FEATURE_NO_TITLE);

        setContentView(R.layout.image_show);
        setTitle("ImageShowActivity");

        mSwitcher = (ImageSwitcher) findViewById(R.id.ImageSwitcher01);
        // 系统的 anim 中的 fade_in.xml
        mSwitcher.setFactory(this);
        mSwitcher.setInAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_in));
        mSwitcher.setOutAnimation(AnimationUtils.loadAnimation(this,
                android.R.anim.fade_out));

        Gallery g = (Gallery) findViewById(R.id.gallery);
        // 为缩略图浏览器指定一个适配器
        g.setAdapter(new ImageAdapter(this));
        // 响应在缩略图列表上选中某个缩略图后的事件
        g.setOnItemSelectedListener(this);

    }
```



```
        @SuppressWarnings("unchecked")
        public void onItemSelected(AdapterView parent, View v, int position, long id) {
                //实现循环显示图片（通过取余的方法来循环获得图像资源 ID）
                mSwitcher.setImageResource(mImageIds[position % mImageIds.length]);
        }

        @SuppressWarnings("unchecked")
        public void onNothingSelected(AdapterView parent) {
        }

        /**
         *ImageSwitcher 组件需要这个方法来创建一个 View 对象（一般为 ImageView 对象）
         *来显示图像
         */
        @Override
        public View makeView() {
                ImageView i = new ImageView(this);
                i.setBackgroundColor(0xFF000000);
                i.setScaleType(ImageView.ScaleType.FIT_CENTER);
                i.setLayoutParams(new ImageSwitcher.LayoutParams(
                                LayoutParams.FILL_PARENT, LayoutParams.FILL_PARENT));
                return i;
        }
}
```

代码清单 11-11　ImageSwitcher+Gallery 使用示例（第 11 章\Demo_11_03）ImageAdapter.java

```
package com.chen.android;

import android.content.Context;
import android.view.View;
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.Gallery;
import android.widget.ImageView;
import android.widget.RelativeLayout.LayoutParams;
/**
 * 图片适配器
 */
public class ImageAdapter extends BaseAdapter {
        private Context mContext;

        /*定义图片数据源*/
        private Integer[] mThumbIds = { R.drawable.f1,
                        R.drawable.f2, R.drawable.f3, R.drawable.f4 };

        public ImageAdapter(Context c) {
                mContext = c;
        }

        // 改进 1，返回最大值（Integer.MAX_VALUE）
```

```
        public int getCount()
        {
        return Integer.MAX_VALUE;
        }

        public Object getItem(int position) {
            return position;
        }
        public long getItemId(int position) {
            return position;
        }
        //getView 方法动态生成一个 ImageView,然后利用 setLayoutParams、setImageResource
        //setBackgroundResource 分别设定图片大小、图片源文件和图片背景。当图片被显示到当前
        //屏幕的时候，这个函数就会被自动回调来提供要显示的 ImageView
        public View getView(int position, View convertView, ViewGroup parent) {

            ImageView i = new ImageView(mContext);
            //i.setImageResource(mThumbIds[position]);
            // 改进 2，通过取余来循环取得 resIds 数组中的图像资源 ID
            i.setImageResource(mThumbIds[position % mThumbIds.length]);
            i.setAdjustViewBounds(true);
            i.setLayoutParams(new Gallery.LayoutParams(
                    LayoutParams.WRAP_CONTENT, LayoutParams.WRAP_CONTENT));
            i.setBackgroundResource(R.drawable.bg0);
            return i;
        }
    }
```

最后，需要在 AndroidManifest.xml 要添加访问权限，如代码清单 11-12 所示。

代码清单 11-12　ImageSwitcher+Gallery 使用示例（第 11 章\Demo_11_03）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

     <application android:label="@string/app_name">
         <activity android:name="MainActivity">
              <intent-filter>
                   <action android:name="android.intent.action.MAIN"></action>
                   <category android:name="android.intent.category.LAUNCHER">
                    </category>
              </intent-filter>
         </activity>
         <activity android:name=".ImageShowActivity"
                   android:label="@string/app_name">
         </activity>
```



```
</application>
</manifest>
```

11.2　订购成功实现

将乘客订票信息提交到后台 ASP 处理后，需要返回预订是否成功的信息。预订成功不等于已经出票（机票行业术语），还需要等待支付后方可出票。支付的方式有现金支付和手机电子支付两种。目前电子支付的安全性及普及性不是很高，机票订购不是小额数目且电子支付需要涉及第三方平台，在这里就不深入讲解。本项目采用的是目前当当网受用户喜爱的货到付款方式。

为了让系统更具人性化，当预订成功后需要以短信的方式通知用户预订的情况，以及什么时候会送票，送票上门时需要支付的金额等信息。

11.2.1　机票订购成功View实现

如第 2 章中的手机订票系统实现效果图（如图 2-15）所示，机票预订功能的 View 实现采用 LinearLayout 布局，是项目的收尾页面，实现起来相对简单。见代码清单 11-13、11-14。

代码清单 11-13　机票订购成功View实现（第 11 章\GanaSkyForTicketOrderSuccess）ordersuccess.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout android:id="@+id/ordersuccess"
      xmlns:android="http://schemas.android.com/apk/res/android"
       android:layout_width="fill_parent"
       android:layout_height="fill_parent"
       android:textSize="15sp"
       android:background="@drawable/buttonbgon"
       android:orientation="vertical">
       <TextView
                  android:id="@+id/ordersuccess_result"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginTop="5dip"
                  android:layout_marginLeft="5dip"/>
         <LinearLayout
                  android:background="@drawable/buttonbgon"
                  android:orientation="horizontal"
                  android:layout_width="fill_parent"
                  android:layout_height="wrap_content"
                  android:layout_marginTop="50dip" >
         <Button
                  android:id="@+id/ordersuccess_btnReturn"
                  android:text="返回查票"
                  android:layout_width="80dip"
                  android:layout_height="40dip"
                  android:layout_marginLeft="15dip"
```




```
                    android:gravity="center"
                    />
            <Button
                    android:id="@+id/ordersuccess_exit"
                    android:text="退出系统"
                    android:layout_width="80dip"
                    android:layout_height="40dip"
                    android:layout_marginLeft="20dip"
                    android:gravity="center"
                    />
            </LinearLayout>
    </LinearLayout>
```

代码清单 11-14 机票订购成功状态栏提示（第 11 章\GanaSkyForTicketOrderSuccess）notify.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
     xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     android:background="@drawable/buttonbgon">

     <TextView
          android:id="@+id/tv_result"
          android:layout_width="fill_parent"
           android:layout_height="fill_parent"
           android:text="谢谢使用 GanaSky 系统" />

</LinearLayout>
```

11.2.2 机票订购成功功能Control实现

机票订购功能 Control 实现，就是机票订购功能的 Activity.java 代码实现，如代码清单 11-15、11-16 所示，基本没有涉及知识难点。在实现过程中需要注意的有以下几点：

1）提供两个按钮，一个按钮返回机票查询界面（或者航班信息界面），涉及页面切换这一知识点；另一个按钮退出系统，退出按钮的代码实现和程序主界面退出功能的代码实现相同。

2）在退出系统的时候，最好将保存的用户信息删除，增强用户个人信息的安全。

3）在状态栏中增加消息提示功能，以便用户查看预订机票的实际状态。

代码清单 11-15 机票订购成功Control实现（第 11 章\GanaSkyForTicketOrderSuccess）OrderSuccess.java（详见本书源代码）

代码清单 11-16 机票订购成功状态栏提示Control实现（第 11 章\GanaSkyForTicketOrderSuccess）OrderSuccessNotify.java（详见本书源代码）

最后，讲解整个项目的配置文件，如代码清单 11-17 所示。

代码清单 11-17 手机订票系统（第 11 章\GanaSkyForTicketOrderSuccess）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
```



```
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ganasky" android:versionCode="1" android:versionName="1.0">

    <application android:icon="@drawable/icon" android:label="@string/app_Iconname"
        android:textSize="15sp">
        <!-- -主页面，对应的是 GanaSkyMain.java 文件  -->
        <activity android:name="GanaSkyMain" android:label="@string/app_ganaSkyMain">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
        <!-- -登录页面  -->
        <activity android:name="LoginApp" android:label="@string/app_nameLogin"
            />
        <!-- -注册页面  -->
        <activity android:name="RegisterApp" android:label="@string/app_nameRegister"
            />

        <!--  机票查询主界面  -->
        <activity android:name="TicketMainApp" android:label="@string/app_nameTicketMain" />
        <activity android:name="CityListView" android:label="@string/app_nameCitySelect" />

        <!--  选择航班  -->
        <activity android:name="FlightInfoMainApp" android:label="@string/app_nameSelectFlight" />

        <!--  航班详细信息  -->
        <activity android:name="FlightListDetail" android:label="@string/app_nameDetailFlight" />

        <!--  订票  -->
        <activity android:name="OrderTicket" android:label="@string/app_nameOrerTicket" />

        <!--  订票返回结果界面-->
        <activity android:name="OrderSuccess" android:label="@string/app_nameOrerTicket" />

        <activity android:name="OrderSuccessNotify" android:label="@string/app_nameOrerTicket" />
    </application>
    <uses-sdk android:minSdkVersion="8" />
    <uses-permission android:name="android.permission.INTERNET" />
    <uses-permission android:name="android.permission.RESTART_PACKAGES" />
    <uses-permission android:name="android.permission.ACCESS_NETWORK_STATE"></uses-permission>
</manifest>
```

第 12 章　界面优化、程序发布与真机环境测试

项目的优化、组装和真机测试，可以看做是项目的总结，其中的每个环节都不容忽视，需要认真对待。本章介绍了如何美化界面，如何进行自适应处理，如何设置程序 Logo 以及如何进行 Android 测试，以进一步完善项目。

12.1　界面效果优化

学习到这里大家会发现，我们所完成的项目的按钮样式效果与第 2 章显示的效果图不一样，这主要是因为没有对按钮进行美化。因此，这一章主要实现界面按钮的美化功能。还可以根据实际需要对项目的其他界面进行美化，有兴趣的读者可以根据自己的需要进行相应的操作。

具体实现步骤如下：

1）在项目 res 下的 drawable 文件夹内新建 buttonshape.xml 和 buttonshape_down.xml 文件，如图 12–1 所示。

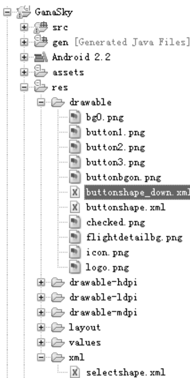


图 12–1　buttonshape.xml 和 buttonshape_down.xml 文件在项目中的位置

buttonshape.xml 是鼠标移出（正常时）的样式，如代码清单 12–1 所示。

代码清单 12–1　按钮美化（第 12 章\GanaSky\res\drawable）buttonshape.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape xmlns:android="http://schemas.android.com/apk/res/android"
      android:shape="rectangle">

     <!--  设置按钮初始画面渐进色  -->
      <gradient android:startColor="#0f61ad" android:endColor="#81F7F3"
                   android:angle="270"/>
        <!--  设置按钮初始画面形状  -->
      <corners
              android:bottomRightRadius="10dip"
              android:bottomLeftRadius="1dip"
              android:topLeftRadius="1dip"
              android:topRightRadius="10dip"/>
</shape>
```

buttonshape_down.xml 是鼠标按下时的样式，如代码清单 12–2 所示。



代码清单 12−2　按钮美化（第 12 章\GanaSky\res\drawable）buttonshape_down.xml

```
<?xml version="1.0" encoding="utf-8"?>
<shape
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:shape="rectangle">

     <gradient
          android:startColor="#F5F510"
          android:endColor="#F5F5B0"
          android:angle="270" />

     <corners
          android:bottomRightRadius="10dip"
          android:bottomLeftRadius="1dip"
          android:topLeftRadius="18dip"
          android:topRightRadius="10dip" />

</shape>
```

2）在项目的 res 目录下新建 xml 文件夹，在文件夹内建 selectshape.xml 文件，如图 12−2 所示。

selectshape.xml 文件是控制按钮正常样式（鼠标移出）鼠标按下时的样式总文件，它可作为按钮背景加入到按钮控件中，如代码清单 12−3 所示。

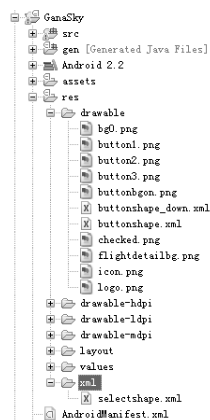


图 12−2　selectshape.xml 文件在项目中的位置

代码清单 12−3　Button按钮美化（第 12 章\GanaSky\res\xml）selectshape.xml

```
<?xml version="1.0" encoding="utf-8"?>
<selector xmlns:android="http://schemas.android.com/apk/res/android">
     <item
          android:state_pressed="false"
          android:drawable="@drawable/buttonshape" />
     <item
          android:state_pressed="true"
          android:drawable="@drawable/buttonshape_down" />
     <item
          android:state_window_focused="false"
          android:drawable="@drawable/buttonshape" />
</selector>
```

经过以上两步的操作，就可以为项目的所有按钮添加 selectshape.xml 样式了。下面以登录按钮的示例代码为例，讲述在 login.xml 中进行按钮美化的过程，如代码清单 12−4 所示。具体做法是在 background 属性中引用该样式在按钮代码中增加：android:background="@xml/selectshape"。

代码清单 12−4　login.xml按钮美化实现

```
<LinearLayout
          android:orientation="horizontal"
```

```
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        android:layout_weight="1.0"
        android:gravity="center_horizontal">
        <Button
              android:id="@+id/btn_Ok"
              android:layout_width="80dip"
                android:layout_height="40dip"
                android:text="确定"
                android:background="@xml/selectshape">
        </Button>
       <Button
              android:id="@+id/btn_Cancel"
              android:layout_width="80dip"
                android:layout_height="40dip"
                android:layout_marginLeft="15dip"
                android:text="取消"
                android:background="@xml/selectshape">
          </Button>
       </LinearLayout>
```

12.2　程序自适应处理

如何在屏幕大小不同的手机上正常显示内容呢？这就要对所定义的控件单位作自适应处理，自适应的关键是如何读取手机屏幕的宽度和高度，程序控件根据手机屏幕大小作自适应处理。下面通过一个示例讲解屏幕上定义的控件如何自适应显示数据，示例代码运行结果如图 12-3 和图 12-4 所示。

图 12-3　ListView 自适应实现表格
——Android 2.2 （QVGA）屏幕

图 12-4　ListView 自适应实现表格
——Android 2.2 （HVGA）屏幕

首先，介绍如何通过 xml 布局实现自适应效果，如代码清单 12-5 所示。

代码清单 12-5　程序自适应示例（第 12 章\Demo_12_01）　main.xml

```
    <?xml version="1.0" encoding="utf-8"?>
```



```
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
     android:layout_height="fill_parent">

    <HorizontalScrollView
      android:id="@+id/HorizontalScrollView01"
      android:layout_height="fill_parent"
      android:layout_width="fill_parent">

         <ListView android:id="@+id/ListView01"
         android:layout_height="wrap_content"
        android:layout_width="wrap_content">
        </ListView>

</HorizontalScrollView>

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 ListView 控件自适应显示内容，如代码清单 12-6、12-7 所示。

代码清单 12-6　程序自适应示例（第 12 章\Demo_12_01）MainActivity.java

```
package com.chen.android;

import java.util.ArrayList;
import com.chen.android.TableAdapter.TableCell;
import com.chen.android.TableAdapter.TableRow;
import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.AdapterView;
import android.widget.ListView;
import android.widget.LinearLayout.LayoutParams;
import android.widget.Toast;

/**
 * ListView 自适应示例
 *
 * @author
 *
 */
public class MainActivity extends Activity {

     ListView lv;
     @Override
     public void onCreate(Bundle savedInstanceState) {

          super.onCreate(savedInstanceState);
          setContentView(R.layout.main);
          this.setTitle("ListView 自适应实现表格");
          //获取布局控件
          lv = (ListView) this.findViewById(R.id.ListView01);
          ArrayList<TableRow> table = new ArrayList<TableRow>();
          //设置表格单元格，每行 5 个单元
```

```
            TableCell[] titles = new TableCell[5];
            //获取手机实际屏幕的宽
            int width = this.getWindowManager().getDefaultDisplay().getWidth()/titles.length;

            // 定义标题
            for (int i = 0; i < titles.length; i++) {
                  titles[i] = new TableCell("标题" + String.valueOf(i), width + 8 * i,
                  LayoutParams.FILL_PARENT, TableCell.STRING);
            }
          //添加表格标题
            table.add(new TableRow(titles));

            // 添加表格每行的数据
            TableCell[] cells = new TableCell[5];// 每行 5 个单元
            for (int i = 0; i < cells.length - 1; i++) {
                  cells[i] = new TableCell("No." + String.valueOf(i),
                  titles[i].width, LayoutParams.FILL_PARENT, TableCell.STRING);
            }

            cells[cells.length - 1] = new TableCell(R.drawable.icon,
            titles[cells.length - 1].width, LayoutParams.WRAP_CONTENT,TableCell.IMAGE);

            // 把表格中每一行的数据添加到表格
            for (int i = 0; i < 12; i++)
           table.add(new TableRow(cells));
          //设置数据适配器
            TableAdapter tableAdapter = new TableAdapter(this, table);
            lv.setAdapter(tableAdapter);
            //监听 listview 子项选择事件
            lv.setOnItemClickListener(new ItemClickEvent());

      }

      /**
       * 子项单击实现
       */
      class ItemClickEvent implements AdapterView.OnItemClickListener {

            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,long arg3) {
               //提示框显示信息
                  Toast.makeText(MainActivity.this,
                              "选中第" + String.valueOf(arg2) + "行", 500).show();
            }
      }

}
```

代码清单 12-7　程序自适应示例（第 12 章\Demo_12_01）ImageAdapter.java

```
package com.chen.android;

import java.util.List;
import android.content.Context;
import android.graphics.Color;
import android.view.Gravity;
import android.view.View;
```



```
import android.view.ViewGroup;
import android.widget.BaseAdapter;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;

/*
 *数据适配器
 */
public class TableAdapter extends BaseAdapter {
   /*定义上下文*/
     private Context context;
     /*实例化多定义的数据表格*/
     private List<TableRow> table;

     public TableAdapter(Context context, List<TableRow> table) {
          this.context = context;
          this.table = table;
     }
  //获取表格的行数
     @Override
     public int getCount() {
          return table.size();
     }
   //获取子项 ID
     @Override
     public long getItemId(int position) {
          return position;
     }
      //根据 ID 查找到子项
     public TableRow getItem(int position) {
          return table.get(position);
     }

     public View getView(int position, View convertView, ViewGroup parent) {
          TableRow tableRow = table.get(position);
          return new TableRowView(this.context, tableRow);
     }

     /**
      *
      * TableRowView 实现表格行的样式
      *
      * @author
      */
     class TableRowView extends LinearLayout {
          public TableRowView(Context context, TableRow tableRow) {

               super(context);
               this.setOrientation(LinearLayout.HORIZONTAL);
               for (int i = 0; i < tableRow.getSize(); i++) {//  逐个格单元添加到行
                    TableCell tableCell = tableRow.getCellValue(i);
                    LinearLayout.LayoutParams layoutParams
                                             = new LinearLayout.LayoutParams(
                    tableCell.width, tableCell.height);//  按照格单元指定的大小设置空间
```




```
                    layoutParams.setMargins(0, 0, 1, 1);// 预留空隙制造边框
                    if (tableCell.type == TableCell.STRING) {// 如果格单元是文本内容
                        TextView textCell = new TextView(context);
                        textCell.setLines(1);
                        textCell.setGravity(Gravity.CENTER);
                        textCell.setBackgroundColor(Color.BLACK);// 背景黑色
                        textCell.setText(String.valueOf(tableCell.value));
                        addView(textCell, layoutParams);
                    } else if (tableCell.type == TableCell.IMAGE) {// 如果格单元是图像内容
                        ImageView imgCell = new ImageView(context);
                        imgCell.setBackgroundColor(Color.BLACK);// 背景黑色
                        imgCell.setImageResource((Integer) tableCell.value);
                        addView(imgCell, layoutParams);
                    }
                }
                // 背景白色，利用空隙来实现边框
                this.setBackgroundColor(Color.WHITE);
            }
        }

        /**
         *
         * TableRow 实现表格的行
         *
         * @author
         */
        static public class TableRow {

            private TableCell[] cell;
            public TableRow(TableCell[] cell) {
                this.cell = cell;
            }

            public int getSize() {
                return cell.length;
            }

            public TableCell getCellValue(int index) {

                if (index >= cell.length)
                    return null;
                return cell[index];
            }

        }

        /**
         *
         * TableCell 实现表格的格单元
         *
         * @author
         */
        static public class TableCell {

            static public final int STRING = 0;
            static public final int IMAGE = 1;
            public Object value;
```



```
            public int width;
            public int height;
            private int type;

            public TableCell(Object value, int width, int height, int type) {
                this.value = value;
                this.width = width;
                this.height = height;
                this.type = type;
            }
        }

    }
```

12.3　设置程序Logo

如何自定义所完成项目的 Logo 图标呢？见代码清单 12-8。具体设置步骤如下。

把制作好的 Logo 图标放到 res\drawable 文件夹里，若需要分不同分辨率显示图片，则需要制作不同分辨率的图片并依次放入 drawable-hdpi、drawable-ldpi 和 drawable-mdpi 文件夹里。

修改 AndroidManifest.xml 配置文件里的内容。android:icon="@drawable/icon"是程序显示图标，icon 是图片名称；android:label="@string/app_Iconname"是图标标题；android:label="@string/app_ganaSkyMain"是程序名称。图标标题和程序名称都引用自 res\ values 下的 strings.xml 文件里的字符串节点内容。

代码清单 12-8　　设置项目的Logo

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
    package="com.ganasky" android:versionCode="1" android:versionName="1.0">

    <application android:icon="@drawable/icon" android:label="@string/app_Iconname"
        android:textSize="15sp">
        <!-- -主页面,对应的是GanaSkyMain.java文件 -->
        <activity android:name="GanaSkyMain" android:label="@string/app_ganaSkyMain">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
```

12.4　Android单元测试

在 Android 开发中，实现单元测试并不很困难，主要是在配置文件和测试环境上需要花大量的时间。Android 单元测试大致可以通过以下几个步骤来实现：

1）新建测试项目。

2）在项目中添加测试单元代码，代码继承自 AndroidTestCase 类。

3）自定义 TestSuite 类，用来管理测试单元代码。

4）自定义 TestRunner 类，用来执行测试。

5）在配置文件 AndroidManifest.xml 中需要增加权限，允许使用单元测试。

进行 Android 单元测试，所涉及的类以及接口如表 12-1 所示。

表 12-1　Android 进行单元测试所涉及的类及接口

Android 进行单元测试所涉及的类及接口	方 法 说 明
AndroidTestCase	Android 单元测试需要从这个类中派生出来，而不再是从 junit.framework.TestCase 中得到。二者之间的最主要区别就是 AndroidTestCase 提供了一个方法 getContext()来获取当前的上下文变量，这个方法在 Android 测试中很重要，因为很多的 Android API 都需要 context 主要成员： setUp()　//创建中间件，如可打开网络连接 tearDown()　//拆掉中间件，如可关闭网络连接 testAndroidTestCaseSetupProperly()　//测试 AndroidTestCase 是否正确装载
TestSuite	在 junit.package 包中，一个 TestSuite 就是一系列测试单元（AndroidTestCase）的集合。通过 TestSuite 可以更好地管理测试单元 TestSuite 主要成员： void addTest (Test　test)　//增加一个测试内容到测试集合里 void addTestSuite(Class testClass)　//从类中增加一系列的测试内容到测试集合中
TestListener	在 junit.framework 包中，这是一个测试接口，用来监听测试进程 有以下 4 个公共方法： abstract void addError(Test test，Throwable t)　// 当测试发生错误时调用 abstract void addFailure(Test test，AssertionFailedError t)　// 当测试失败时调用 abstract void endTest(Test test)　// 当测试结束时调用 abstract void startTest(Test test)　//当测试开始时调用
AndroidTestRunner	继承自 class junit.runner.BaseTestRunner 类，但是它没有提供 UI， 甚至是基于控制台的 UI 都没有，因此，如果想要很好地查看测试结果，需要自定义处理类来进行相应的操作，这个处理类需要使用到 AndroidTestRunner 中的 callback 函数 AndroidTestRunner 类的主要方法： setTest();　//设置测试 runTest();　//运行测试 addTestListener();　//增加测试监听 setContext();　//设置上下文变量

如何通过编写代码进行单元测试呢？具体过程如下：

1）第一步，新建一个 TestCase 测试项目（为了方便程序管理，本例取名为 Demo_12_02），在项目中建 TestContent 类，这个类需要继承 androidTestCase，才能通过 getContext()来获取当前的上下文变量。具体流程是：单击菜单栏”文件”→“新建”→“项目”→“Android”→“Android Test　Project”选择，如图 12-5 至图 12-7 所示。



图 12-5　新建 Android 测试项目界面（1）



图 12-6　新建 Android 测试项目界面（2）



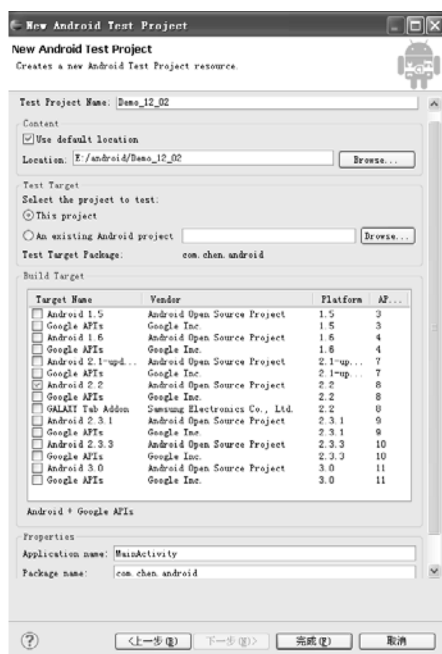


图 12-7　新建 Android 测试项目界面（3）

2）第二步，在项目中新建 AndroidTestCase 类。它需要继承自 AndroidTestCase 类才能完成单元测试。见代码清单 12-9。

代码清单 12-9　Android单元测试示例（第 12 章\Demo_12_02）TestContent.java

```
package com.chen.android;

import android.test.AndroidTestCase;
/**
* 测试内容类
 * @author
 *
 */
public class TestContent extends AndroidTestCase {

    private   int i1;
    private int i2;
    static final String LOG_TAG="MyTestCase";

    @Override
    protected void setUp() throws Exception{
        i1=2;
        i2=3;
    }
    /**
     * 所要测试的内容，若(i1+i2)==5 为 true，不抛出 assertTrue 的内容，反之亦然
     */
    public void testAdd(){
        assertTrue("抛出 AssertionFailedError，则 failed=1"，(i1+i2)==5);
```

```
        }

        /**
         * 所要测试的内容，若(i1-i2)==1 为 true，不抛出 assertTrue 的内容，反之亦然
         */
        public void testDec(){
                assertTrue("抛出  AssertionFailedError，则 failed=1", (i1-i2)==1);
        }

        @Override
        protected void tearDown() throws Exception{
                super.tearDown();
        }

        @Override
        public void testAndroidTestCaseSetupProperly(){
                super.testAndroidTestCaseSetupProperly();
        }

}
```

3）第三步，新建一个 ExampleSuite 类，这个类继承 Junit 的 TestSuit，注意在这里要使用 addTestSuite()方法，一开始就使用 addTest()方法不能成功。见代码清单 12-10。

代码清单 12-10　Android单元测试示例（第 12 章\Demo_12_02）ExampleSuite.java

```
package com.chen.android;
public class ExampleSuite extends junit.framework.TestSuite {
   public ExampleSuite(){
           addTestSuite(TestContent.class);
   }
}
```

4）第四步，新建一个 Activity 类，见代码清单 12-11，用来启动单元测试，并显示测试结果。

代码清单 12-11　Android单元测试示例（第 12 章\Demo_12_02）MainActivity.java

```
package com.chen.android;
import junit.framework.Test;
import junit.framework.TestListener;
import android.app.Activity;
import android.graphics.Color;
import android.os.Bundle;
import android.os.Handler;
import android.os.Message;
import android.test.AndroidTestRunner;
import android.view.View;
import android.widget.Button;
import android.widget.TextView;
import android.widget.Toast;
/**
 * 测试主程序
 * @author
```



```
 *
 */
public class MainActivity extends Activity {
   /*定义布局控件*/
     private TextView resultView;

     private TextView barView;

     private TextView messageView;

     private Thread testRunnerThread;
   /*显示测试结果*/
     private static final int SHOW_RESULT = 0;
     /*显示测试错误*/
     private static final int ERROR_FIND = 1;

     @Override
     protected void onCreate(Bundle savedInstanceState) {

          super.onCreate(savedInstanceState);
          /*获取布局控件*/
          setContentView(R.layout.main);
          resultView = (TextView) findViewById(R.id.resultView);
          barView = (TextView) findViewById(R.id.barView);
          messageView = (TextView) findViewById(R.id.messageView);
          Button lunch = (Button) findViewById(R.id.lunchButton);

          lunch.setOnClickListener(new View.OnClickListener() {
               @Override
               public void onClick(View v) {
                    /*开始测试*/
                    startTest();
               }
          });
     }

     /*显示测试的错误个数*/
     private void showMessage(String message) {
          hander.sendMessage(hander.obtainMessage(ERROR_FIND, message));
     }

     /*显示测试结果*/
     private void showResult(String text) {
          hander.sendMessage(hander.obtainMessage(SHOW_RESULT, text));
     }

     /*开始测试*/
     private synchronized void startTest() {
          if (testRunnerThread != null && testRunnerThread.isAlive()) {
               testRunnerThread = null;
          }
          if (testRunnerThread == null) {
               testRunnerThread = new Thread(new TestRunner(this));
               testRunnerThread.start();
          } else {
```




```
                Toast.makeText(this, "Test is still running", Toast.LENGTH_SHORT)
                        .show();
            }
        }
        /*定义线程*/
        public Handler hander = new Handler() {
            @Override
            public void handleMessage(Message msg) {
                switch (msg.what) {
                case SHOW_RESULT:
                    resultView.setText(msg.obj.toString());
                    break;
                case ERROR_FIND:
                    messageView.append(msg.obj.toString());
                    barView.setBackgroundColor(Color.RED);
                    break;
                default:
                    break;
                }
            }
        };
    /**
     * 运行测试类
     * @author
     *
     */
        class TestRunner implements Runnable, TestListener {

            private Activity parentActivity;
            private int testCount;
            private int errorCount;
            private int failureCount;

            public TestRunner(Activity parentActivity) {
                this.parentActivity = parentActivity;
            }

            @Override
            public void run() {
                testCount = 0;
                errorCount = 0;
                failureCount = 0;

                ExampleSuite suite = new ExampleSuite();
                AndroidTestRunner testRunner = new AndroidTestRunner();
                testRunner.setTest(suite);
                testRunner.addTestListener(this);
                testRunner.setContext(parentActivity);
                testRunner.runTest();
            }

            @Override
```



```
            public void addError(Test test, Throwable t) {
                errorCount++;
                showMessage(t.getMessage() + "\n");
            }
            @Override
            public void endTest(Test test) {
                showResult(getResult());
            }
            @Override
            public void startTest(Test test) {
                testCount++;
            }
            private String getResult() {
                int successCount = testCount - failureCount - errorCount;
                return "Test:" + testCount + " Success:" + successCount
                        + " Failed:" + failureCount + " Error:" + errorCount;
            }
            @Override
            public void addFailure(Test test, junit.framework.AssertionFailedError t) {
                // TODO Auto-generated method stub
                failureCount++;
                showMessage(t.getMessage() + "\n");

            }

        }

    }
```

在 AndroidManifest.xml 配置文件中加入 Android.test.runner 权限，见代码清单 12-12，否则无法进行单元测试。

代码清单 12-12　Android单元测试示例（第 12 章\Demo_12_02）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />
    <instrumentation android:targetPackage="com.chen.android"
      android:name="android.test.InstrumentationTestRunner" />
    <application android:icon="@drawable/icon" android:label="@string/app_name">

    <uses-library android:name="android.test.runner" />
    </application>
</manifest>
```

代码运行结果如图 12-8 所示。

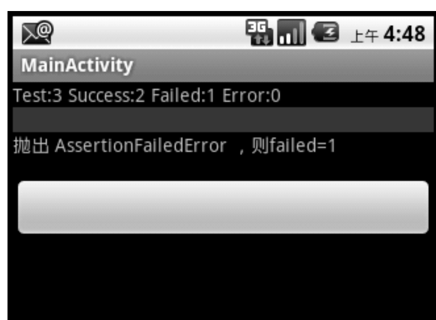


图 12-8 Android 单元测试

12.5 项目在不同版本的Android手机上的安装与运行

这是项目的最后一步，是如何将开发的程序打包成不同版本的 Android.apk 文件的过程。Android 有不同版本的操作系统，如何在不同 Android 手机操作系统上都能成功安装所开发的项目呢？这需要修改 AndroidManifest.xml 中的 android:minSdkVersion 值，例如：android:minSdkVersion="8"，表示支持 2.2 的 Android 手机操作系统。

Android 程序在运行过程中，会自动编译生成.apk 文件，该文件可以在 Android 手机操作系统被执行，.apk 文件存放在项目的 bin 目录下。在手机上安装 Android 程序的具体步骤详见附录 B。

第 13 章　其他Andorid专题开发

手机订票系统项目开发基本完成，涵盖了几乎所有应用型控件的使用，强调的是 Webservice/Mobile 手机开发模式。但有少部分的手机软件无需与后台进行通信，无需在远程服务器上建立数据库，只需在本地手机上建一个小型数据库实现这一类型的软件就要学会如何在 Android 手机中存储数据，读取手机本身的资源文件，这一技术的实现重点就是数据存储问题。为了解决这一问题，Android 提供了多种方式的数据存储，本章将重点讲解 Android 数据存储。在本章中还讲解了 Android 多媒体开发及外部接口编程这两个 Android 专题开发。

13.1　专题一：Andorid数据存储

Android 开发架构中一共提供了五种方式的数据存储，它们分别是：SharedPreferences、FilesShared 、SQLite、Network 及 ContentProviders 方式。SharedPreferences 主要用来存储"Key"-"Value"格式的数据，是一个轻量级的键值存储机制，只可以存储基本数据类型；Filesshared 属于文件存储，就是把需要保存的文件存放到本地资源文件夹中（如 sdcard 中），需要获取数据时则向文件夹里读取；SQLite 是 Android 提供的标准数据库，支持 SQL 语句。它是一个开源的关系数据库，可以存储大量数据，并且可以很容易的进行增、删、改、查；Network 则是通过网络来存储和获取数据。以上这几种方式保存数据，都是应用程序私有的，如果需要在其他应用程序中使用这些数据，就要使用 Android 提供的 ContentProviders（数据共享），此种类型的数据存储允许其他应用程序访问。SharedPreferences 存储在第 4 章中已详细介绍，这里就不再赘述。

13.1.1　Files存储

同前面所使用的 SharedPreferences 一样，使用 File 来读写文件也属于常规思路。它是一个应用程序私有的，一个应用程序无法读写其他应用程序的文件。在 Android 中没有提供像 J2SE 里面那么多关于 IO 的 API，所以使用起来非常简单轻巧。Files 方式的存储是把需要保存的数据通过文件的形式记录下来，当需要这些数据时，通过读取这个文件并获得相应的数据。大家都知道 Android 采用的是 Linux 内核，所以在 Android 系统中，文件的形式与 Linux 中的文件形式相同。

在实际开发中如何通过 File 读取和保存数据呢？可以使用手机本身的存储设备或者其他的外接存储设备创建用于保存数据的文件。具体是通过 FileInputStream 和 FileOutputStream 文件数据流对文件进行操作，可以通过 openFileOutput 函数打开一个文件，如果文件不存在就自动创建，通过 load()方法来获取文件中的数据，通过 store()方法来获取文件中的数据，通过 deleteFile()方法可以删除一个指定的文件。在 Android 系统中，这些文件保存在“/data/data/PACKAGE_NAME（所建的包名）/files”目录下，见代码清单 13-1。数据读取

和保存的代码实现如下：

代码清单 13-1　File方式数据读取

```
/**
 * 读取数据（简单方法）
 *
 * @param context
 * @param file
 * @return
 */
public static String read(Context context, String file) {
    String data = "";
    try {
        FileInputStream stream = context.openFileInput(file);
        StringBuffer sb = new StringBuffer();
        int c;
        while ((c = stream.read()) != -1) {
            sb.append((char) c);
        }
        stream.close();
        data = sb.toString();
    } catch (FileNotFoundException e) {
    } catch (IOException e) {
    }
    return data;
}
```

从代码中进一步看出，通过 context.openFileInput(file)代码打开文件，Android 中的文件读写具有权限的控制，因此要使用 context()来打开文件，文件可以在相同的 Package 中共享，见代码清单 13-2。

代码清单 13-2　File方式数据写入

```
/**
 * 写数据（简单方法）
 * @param context
 * @param file
 * @param msg
 */
public static void write(Context context, String file, String msg) {
    try {
        FileOutputStream stream = context.openFileOutput(file,
                Context.MODE_WORLD_WRITEABLE);
        stream.write(msg.getBytes());
        stream.flush();
        stream.close();
    } catch (FileNotFoundException e) {
    } catch (IOException e) {
    }
}
```



从代码清单 13-2 可以看出，它声明了文件打开的方式：context.openFileOutput(file, Context.*MODE_WORLD_WRITEABLE*)。一般来说，直接使用文件可能不太好用，尤其是存放一些琐碎的数据，会生成一些琐碎的文件，且由于格式不同不便于管理，因此可以将需要保存的数据包装成 Properties 属性来使用，具体实现见本章示例代码。

下面通过一个示例讲解 File 方式数据存储的过程，代码运行结果如图 13-1 所示。当重新启动的时候，在用户名编辑框中出现上一次保存的数据。



图 13-1　File 方式数据存储示例

首先，介绍如何通过 xml 布局实现这一效果，如代码清单 13-3 所示。

代码清单 13-3　File方式数据存储示例（第 13 章\Demo_13_01）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
  <TextView
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="用户名"
    />
  <EditText
      android:id="@+id/txtUserName"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:maxLength = "20"
      />
  <TextView
       android:layout_width="fill_parent"
       android:layout_height="wrap_content"
       android:text="密码"
       />
  <EditText
      android:id="@+id/txtPass"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
      android:maxLength = "20"
      android:password = "true"
      />
   <Button
      android:id="@+id/btnSend"
      android:layout_width="fill_parent"
      android:layout_height="wrap_content"
```

```
            android:text="Flie 方式保存用户信息"
            android:gravity = "center"
            android:width = "80px"
            />

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现 Flies 数据存储功能，如代码清单 13-4 所示。

代码清单 13-4　File方式数据存储示例（第 13 章\Demo_13_01）MainActivity.java

```
package com.chen.android;

import java.io.FileInputStream;
import java.io.FileNotFoundException;
import java.io.FileOutputStream;
import java.io.IOException;
import java.util.Properties;
import android.app.Activity;
import android.os.Bundle;
import android.view.Gravity;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
import android.content.Context;

/**
 * File 方式保存
 *
 * @author
 *
 */
public class MainActivity extends Activity {
    /*对象声明*/
      private Button btnsend;
      private EditText ed_username;
      private EditText ed_pass;
      private Properties properties;
      private String userName = "";
      private String userPwd = "";

      @Override
      public void onCreate(Bundle savedInstanceState) {
            super.onCreate(savedInstanceState);
            setContentView(R.layout.main);
        /*获取对象*/
            btnsend = (Button) this.findViewById(R.id.btnSend);
            ed_username = (EditText) this.findViewById(R.id.txtUserName);
            ed_pass = (EditText) this.findViewById(R.id.txtPass);
          /*加载保存的信息*/
            loginUserInfo();
```



```
        /*保存按钮事件监听*/
        btnsend.setOnClickListener(Save);
    }

    /**
     * 保存用户信息
     */
    private void loginUserInfo() {

        properties = new Properties();
        properties = load(MainActivity.this, "use.xml");
        if (properties.isEmpty()) {

        } else {
            userName = properties.get("usename").toString();
            userPwd = properties.get("usepwd").toString();

            ed_username.setText(userName);
            ed_pass.setText(userPwd);
        }
    }

    /**
     * 保存
     */
    Button.OnClickListener Save = new Button.OnClickListener() {
        @Override
        public void onClick(View v) {
            userName = ed_username.getText().toString();
            userPwd = ed_pass.getText().toString();
            if (!"".equals(userName) && !"".equals(userPwd)) {
                properties = new Properties();
                /* 将数据打包成 Properties */
                properties.put("usename", String.valueOf(userName));
                properties.put("usepwd", String.valueOf(userPwd));

                store(MainActivity.this, "use.xml", properties);

                DisplayToast("保存成功！");

            } else {

            }
        }
    };

    /**
     * 读取数据（较优方法）
     * @param context
     * @param file
     * @return
     */
    public static Properties load(Context context, String file) {
        Properties properties = new Properties();
        try {
            FileInputStream stream = context.openFileInput(file);
```




```
                properties.load(stream);
            } catch (FileNotFoundException e) {
            } catch (IOException e) {
            }
            return properties;
    }

    /**
     * 保存数据（较优方法）
     *
     * @param context
     * @param file
     * @param properties
     */
    public static void store(Context context, String file, Properties properties) {
            try {
                FileOutputStream stream = context.openFileOutput(file,
                            Context.MODE_WORLD_WRITEABLE);
                properties.store(stream, "");
            } catch (FileNotFoundException e) {
            } catch (IOException e) {
            }
    }

    /**
     * 读取数据（简单方法）
     *
     * @param context
     * @param file
     * @return
     */
    public static String read(Context context, String file) {
            String data = "";
            try {
                FileInputStream stream = context.openFileInput(file);
                StringBuffer sb = new StringBuffer();
                int c;
                while ((c = stream.read()) != -1) {
                    sb.append((char) c);
                }
                stream.close();
                data = sb.toString();
            } catch (FileNotFoundException e) {
            } catch (IOException e) {
            }
            return data;
    }

    /**
     * 写数据（简单方法）
     * @param context
     * @param file
```



```
             * @param msg
             */
            public static void write(Context context, String file, String msg) {
                    try {
                            FileOutputStream stream = context.openFileOutput(file,
                                            Context.MODE_WORLD_WRITEABLE);
                            stream.write(msg.getBytes());
                            stream.flush();
                            stream.close();
                    } catch (FileNotFoundException e) {
                    } catch (IOException e) {
                    }
            }

            /*  显示 Toast */
            public void DisplayToast(String str) {
                    Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
                    //  设置 toast 显示的位置
                    toast.setGravity(Gravity.TOP, 0, 220);
                    //  显示该 toast
                    toast.show();
            }

    }
```

13.1.2　NetWork存储

NetWork 存储是通过网络来保存和获取数据资源，这种方式的数据存储需要手机设备保持网络连接状态。将数据存储到网络上的方法有很多，比如将要保存的数据以文件的方式上传到服务器上，还可以通过 NetWork 发送邮件。下面通过一个示例讲述 NetWork 如何获取网络上的资源，如何发送邮件的过程。要使用 NetWork 发送邮件这一功能就要学会如何发送电子邮件，这首先需要在模拟器中配置电子邮件账户，下面介绍模拟器电子邮件账户的配置过程。

1）切换到 Android 模拟器的主菜单，单击“Email”，如图 13-2 所示。

2）填写邮箱地址密码，单击下一步，如图 13-3 所示。

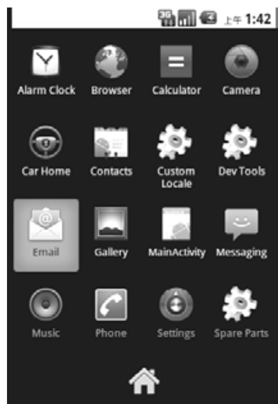


图 13-2　Android 模拟器邮件配置（1）



图 13-3　Android 模拟器邮件配置（2）

3）选择账号类型，在这里选择 POP3 类型. 如图 13-4 所示。

4）配置接收服务器设置，注意 POP3 服务器端类型和端口号的填写，不同的局域网 pop3 服务器端类型设置不同。比如：一般高校的网络配置有限制，需要使用学校邮箱和规定的服务器地址才能配置成功。填写完成，单击下一步，如图 13-5 所示。



图 13-4　Android 模拟器邮件配置（3）

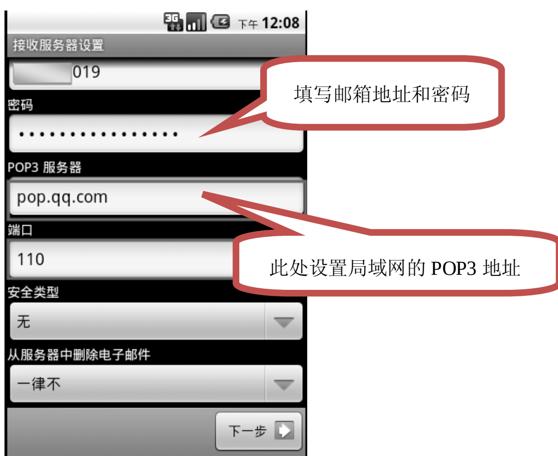


图 13-5　Android 模拟器邮件配置（4）

5）等待接收服务器设置。在这里有时邮箱配置可能失败，弹出无法连接的提示，这就是因为所使用的局域网不支持 POP3 的缘故，如图 13-6 所示。

6）若连接成功，会弹出“外发服务器设置”。SMTP 服务器设置为：smtp.qq.com，则端口设置一定要填成 25。单击下一步，显示正在检测外发服务器配置进度栏，如图 13-7 和图 13-8 所示。



图 13-6　Android 模拟器邮件配置（5）



图 13-7　Android 模拟器邮件配置（6）

如图 13-9 所示，弹出账户选项界面，表示配置成功。最后，还可以为邮箱设置用户名等信息，单击完成，可以看到所配置的邮箱中的邮件信息了。详见图 13-10 至图 13-11。





图 13-8　Android 模拟器邮件配置（7）



图 13-9　Android 模拟器邮件配置（8）



图 13-10　Android 模拟器邮件配置（9）



图 13-11　Android 模拟器邮件配置（10）

配置好模拟器的电子邮件后，就可以通过程序来为用户发邮件了。在示例中，我们捕获键盘上的上方键和下方键，当按上方键时发邮件，按下方键则从网络上下载资源。Android 中发送电子邮件的过程是：通过 startActivity()方法来调用要发送的邮件数据的 Intent，通过 putExtra()方法来设置邮件的主题、内容、附件等信息。示例代码运行结果如图 13-12 至图 13-14 所示。

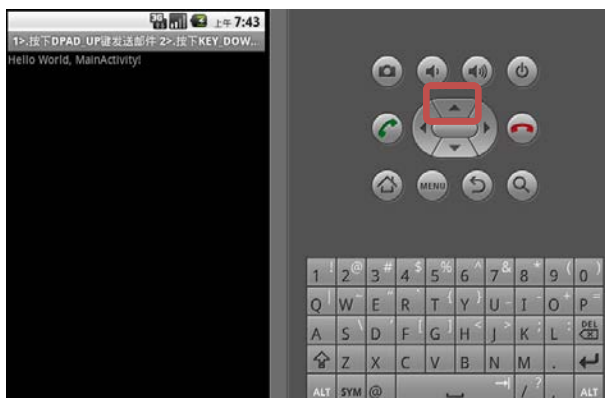


图 13-12　NetWork 方式存储示例代码运行结果（1）——初始化



图 13-13　NetWork 方式存储示例代码运行结果（2）——按上方键发邮件

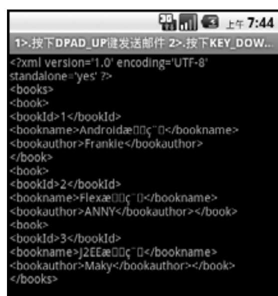


图 13-14　NetWork 方式存储示例代码运行结果（3）——按下方键下载资源

首先，介绍如何通过 xml 布局实现界面效果，如代码清单 13-5 所示。

代码清单 13-5　File方式数据存储示例（第 13 章\Demo_13_02）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent"
     >
<TextView
      android:id="@+id/TextView"
     android:layout_width="fill_parent"
     android:layout_height="wrap_content"
     android:text="@string/hello"
     />

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合 xml 布局实现 NetWork 存储数据这一功能，如代码清单 13-6 所示。

代码清单 13-6　File方式数据存储示例（第 13 章\Demo_13_02）MainActivity.java

```
package com.chen.android;

import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;
import android.app.Activity;
import android.content.Intent;
import android.net.Uri;
import android.os.Bundle;
import android.view.KeyEvent;
import android.widget.TextView;
/**
  * NetWork 存储
  * @author
  *
  */
public class MainActivity extends Activity {
```



```
/*初始化对象*/
 TextView tv;
@Override
public void onCreate(Bundle savedInstanceState) {
    super.onCreate(savedInstanceState);
    setContentView(R.layout.main);
    this.setTitle("按下 DPAD_UP 键发送邮件；按下 KEY_DOWN 键下载网络上的数据");
    tv = (TextView) this.findViewById(R.id.TextView);
}
/**
 * 重写 onKeyDown 按键方法
 */
 @Override
 public boolean onKeyDown(int keyCode, KeyEvent event) {
      //单击上方键发邮件
      if(keyCode==KeyEvent.KEYCODE_DPAD_UP){
           //邮件地址，注意：此处一定要加<mailto:>，否则无法发送
           Uri uri = Uri.parse("mailto:546393019@qq.com");
           //创建 Intent
           Intent intent = new Intent(Intent.ACTION_SENDTO,uri);
           //设置邮件主题
           intent.putExtra(Intent.EXTRA_SUBJECT, "邮件标题：您好！");
           //设置邮件内容
  intent.putExtra(Intent.EXTRA_TEXT, "邮件内容：太高兴了，我学会用 android 发邮件啦！");
           //启动 intent
           this.startActivity(intent);

      //单击下方键下载网络资源
      }else if(keyCode==KeyEvent.KEYCODE_DPAD_DOWN){

           String myString = "";
           try{
                //定义访问的地址
                URL url = new URL("http://220.165.15.216:7777/MobilePage/books.xml");
                //打开网络连接
                URLConnection urlconn = url.openConnection();
                //获得输入流
                InputStream input = urlconn.getInputStream();
                int count = 0;
                //初始化字符串 buffer
                StringBuffer sb = new StringBuffer();
                while((count = input.read())!=-1){
                     sb.append((char)count);
                }
                //将下载的网络资源显示在 TextView 控件上
                myString = sb.toString();
           }catch(Exception ex){
                //捕获异常
                myString = ex.getMessage();
                ex.printStackTrace();

           }
```

```
                tv.setText(myString);
            }
            return super.onKeyDown(keyCode, event);
        }

    }
```

最后，需要在 AndroidManifest.xml 要添加访问权限，具体如代码清单 13-7 所示。

代码清单 13-7　File方式数据存储示例（第 13 章\Demo_13_02）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<manifest xmlns:android="http://schemas.android.com/apk/res/android"
      package="com.chen.android"
      android:versionCode="1"
      android:versionName="1.0">
    <uses-sdk android:minSdkVersion="8" />

    <application android:icon="@drawable/icon" android:label="@string/app_name">
        <activity android:name=".MainActivity"
                  android:label="@string/app_name">
            <intent-filter>
                <action android:name="android.intent.action.MAIN" />
                <category android:name="android.intent.category.LAUNCHER" />
            </intent-filter>
        </activity>
    </application>
   <uses-permission android:name="android.permission.INTERNET" />
</manifest>
```

13.1.3　SQLite编程详解

（1）SQLite 简介

Android 操作系统中内置了 SQLite 数据库，它属于轻量级、嵌入式的关系型数据库，目前已经在 iPhone、Android 等手机系统中使用。SQLite 可移植性相对较好，较容易使用，且具有轻小、高效、安全可靠等特性。其结构如图 13-5 所示。

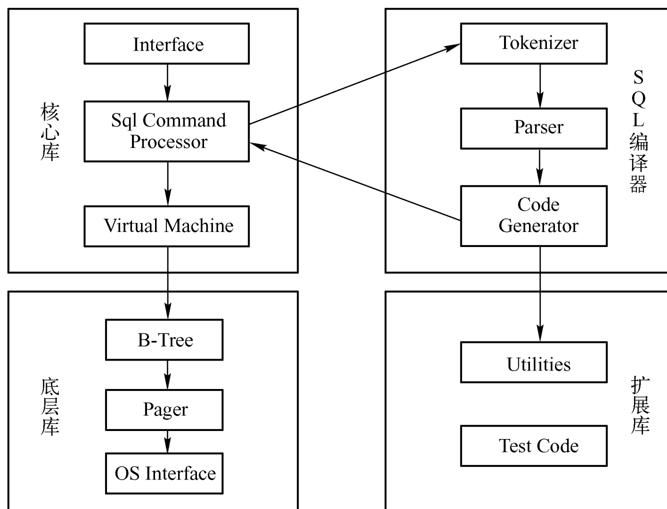


图 13-15　SQLite 结构图



SQLite 由以下几个组件组成：SQL 编译器、内核、后端以及附件。SQLite 通过利用虚拟机和虚拟数据库引擎（VDBE）使调试、修改和扩展 SQLite 的内核变得更加方便。SQLite 嵌入到使用它的应用程序中，它们共用相同的进程空间，而不是单独的一个进程。从外部看，它并不像一个关系型数据库（RDBMS），但在进程内部，它却是完整的，有自包含的数据库引擎。

SQLite 基本上符合 SQL-92 标准，和其他主流的 SQL 数据库没有什么区别。它的优点就是高效，是一个完整的 SQL 系统，拥有完整的触发器、交易等功能。它和其他数据库最大的不同是：对数据类型的支持。当创建一个表时，可以在 CREATE TABLE 语句中指定某列的数据类型，但是可以把任何数据类型放入任何列中；当某个值插入数据库时，SQLite 将检查到该值的类型。如果该类型与关联的列不匹配，则 SQLite 会尝试将该值转换成该列的类型；如果不能转换，则该值将作为其本身具有的类型存储。SQLite 称这一数据类型为“弱类型”（Manifest Typing）。此外，SQLite 不支持一些标准的 SQL 功能，特别是外键约束（FOREIGN KEY constrains）、嵌套 TRANSCACTION 、RIGHT OUTER JOIN 、 FULL OUTER JOIN 以及 ALTER TABLE 等功能。

（2）Android　SQLite

Android 在运行时（Run-Time）集成了 SQLite，所以每个 Android 应用程序都可以使用 SQLite 数据库。对于熟悉 SQL 的开发人员来说，在 Android 开发中使用 SQLite 相当简单。但是，由于 JDBC 会消耗太多的系统资源，所以 JDBC 对于手机这种内存限制较大的设备来说并不合适。因此，Android 提供了一些新的 API 来使用 SQLite 数据库。在 Android 中建数据库，数据库保存的具体位置是在 DDMS 中的 File Explorer 中的“data/data/<项目文件夹>/databases/”下。

下面详细讲解 Android　SQLite 创建数据库，添加数据和查询数据库的过程。

1）SQLite 创建数据库。

```
/* 声明数据库对象 */
	private SQLiteDatabase	mSQLiteDatabase	= null;
/* 打开数据库 */
mSQLiteDatabase = this.openOrCreateDatabase(DATABASE_NAME,

MODE_PRIVATE, null);
```

如果创建不成功则抛出 FileNotFoundException 异常。

2）SQLite 关闭数据库。

```
mSQLiteDatabase.close();
```

3）SQLite 删除指定数据库。

```
this.deleteDatabase("myDate");
```

4）SQLite 执行 SQL 命令。

```
/* 在数据库 mSQLiteDatabase 中创建一个表 */
mSQLiteDatabase.execSQL(CREATE_TABLE);
```

下面通过一个示例巩固 Android SQLite 的具体用法，示例的主要功能包括创建数据

库、删除数据库、创建表、删除表以及对数据进行增删改查的操作等。示例代码运行结果如图 13-16 所示。

首先，介绍如何通过 xml 布局实现界面效果，如代码清单 13-8 所示。



图 13-16 SQLite 存储示例代码运行结果

代码清单 13-8 SQLite 存储示例（第 13 章\Demo_13_03）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
        android:id="@+id/LinearLayout01"
        android:layout_width="fill_parent"
        android:layout_height="fill_parent"
        xmlns:android="http://schemas.android.com/apk/res/android"
        android:orientation="vertical"
        >
       <EditText
               android:id="@+id/txtInput"
               android:layout_width="fill_parent"
               android:layout_height="wrap_content"
               android:text="你好！"
       />

       <LinearLayout
               android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal">
               <Button
                       android:id="@+id/btnAdd"
                       android:layout_width="wrap_content"
                       android:layout_height="wrap_content"
                       android:text="增加子项"
               />
               <Button
                       android:id="@+id/btnDelData"
                       android:layout_width="wrap_content"
                       android:layout_height="wrap_content"
                       android:text="删除子项"
               />
                 <Button
                       android:id="@+id/btnUpdateData"
                       android:layout_width="wrap_content"
                       android:layout_height="wrap_content"
                       android:text="更新数据"
               />
        </LinearLayout>
          <LinearLayout
               android:layout_width="fill_parent"
                android:layout_height="wrap_content"
                android:orientation="horizontal">
                 <Button
```



```
                android:id="@+id/btnDelDataTable"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="删除表"
            />
        <Button
                android:id="@+id/btnDelDataBase"
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"
                android:text="删除数据库"
            />
    </LinearLayout>

    <ListView
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"
        android:id="@+id/ListView01"/>

</LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现 SQLite 存储过程，如代码清单 13-9 所示。

代码清单 13-9　SQLite存储示例（第 13 章\Demo_13_03）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.database.sqlite.SQLiteDatabase;
import android.os.Bundle;
import android.view.KeyEvent;
import android.view.View;
import android.widget.Button;
import android.widget.EditText;
import android.widget.LinearLayout;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
/**
 * SQLite 的使用方法
 * @author
 *
 */
public class MainActivity extends Activity
{
    /* 记录条数*/
    private static int          miCount = 0;

    /* 数据库对象 */
    private SQLiteDatabase          mSQLiteDatabase = null;
    /* 数据库名 */
```




```
        private final static String     DATABASE_NAME = "MyData.db";
        /* 表名 */
        private final static String     TABLE_NAME = "table1";

        /* 表中的字段 */
        private final static String     TABLE_ID = "_id";
        private final static String     TABLE_NUM = "num";
        private final static String     TABLE_DATA = "data";

        /* 创建表的 sql 语句 */
        private final static String     CREATE_TABLE= "CREATE TABLE " +
                                    TABLE_NAME + " (" + TABLE_ID + " INTEGER
PRIMARY KEY," +
                                      TABLE_NUM + " INTERGER,"+ TABLE_DATA + "
TEXT)";
        /* 线性布局 */
        LinearLayout                    m_LinearLayout = null;
        /* 列表视图-显示数据库中的数据 */
        ListView                        m_ListView = null;

       /*初始化控件对象*/
        EditText inputTxt;
        Button btnAdd;
        Button btnDelData;
        Button btnDelDataTable;
        Button btnDelDataBase;
        Button btnUpdateData;

        @Override
        public void onCreate(Bundle savedInstanceState)
        {
            super.onCreate(savedInstanceState);
            /*获取控件*/
            this.setContentView(R.layout.main);
            btnAdd = (Button) findViewById(R.id.btnAdd);
            btnDelData = (Button) findViewById(R.id.btnDelData);
            btnDelDataTable = (Button) findViewById(R.id.btnDelDataTable);
            btnDelDataBase = (Button) findViewById(R.id.btnDelDataBase);
            btnUpdateData=(Button) findViewById(R.id.btnUpdateData);
            inputTxt = (EditText) findViewById(R.id.txtInput);
            m_ListView=(ListView)findViewById(R.id.ListView01);

            // 设置监听
            btnAdd.setOnClickListener(btnAddClick);
            btnDelData.setOnClickListener(DelData);
            btnUpdateData.setOnClickListener(UpdateData);
            btnDelDataTable.setOnClickListener(DelDataTable);
            btnDelDataBase.setOnClickListener(DelDataBase);

            // 打开已经存在的数据库
        mSQLiteDatabase = this.openOrCreateDatabase(DATABASE_NAME,

MODE_PRIVATE, null);
```



```
        try
        {
            /* 在数据库 mSQLiteDatabase 中创建一个表 */
            mSQLiteDatabase.execSQL(CREATE_TABLE);
        }
        catch (Exception e)
        {
            // 获取数据库 Phones 的 Cursor，为 ListView 添加数据源
            UpdataAdapter();
        }
    }

    /**
     * 增加数据
     */
    Button.OnClickListener btnAddClick = new Button.OnClickListener() {

        public void onClick(View v) {
            // 增加
            AddData(inputTxt.getText().toString());
        }
    };

    /**
     * 删除数据
     */
    Button.OnClickListener DelData = new Button.OnClickListener() {

        public void onClick(View v) {
          //删除数据
            DeleteData();
        }
    };

    /**
     * 更新数据
     */
    Button.OnClickListener UpdateData = new Button.OnClickListener() {

        public void onClick(View v) {
           //更新数据
            UpData();
        }
    };
    /**
     * 删除表
     */
    Button.OnClickListener DelDataTable = new Button.OnClickListener() {

        public void onClick(View v) {
           // 删除表
            DeleteTable();
        }
    };
```

```
/**
 * 删除数据库
 */
Button.OnClickListener DelDataBase = new Button.OnClickListener() {

	public void onClick(View v) {
	//删除数据库
		DeleteDataBase();

	}
};

/* 删除数据库 */
public void DeleteDataBase()
{
	this.deleteDatabase(DATABASE_NAME);
	this.finish();
}

/* 删除一个表 */
public void DeleteTable()
{
	mSQLiteDatabase.execSQL("DROP TABLE " + TABLE_NAME);
	this.finish();
}

/* 更新一条数据 */
public void UpData()
{
	ContentValues cv = new ContentValues();
	cv.put(TABLE_NUM, miCount);
	cv.put(TABLE_DATA, "修改后的数据" + miCount);

	/* 更新数据 */
	mSQLiteDatabase.update(TABLE_NAME, cv, TABLE_NUM + "="
									+ Integer.toString(miCount - 1),
null);

	UpdataAdapter();
}

/* 向表中添加一条数据 */
public void AddData(String	data)
{
	ContentValues cv = new ContentValues();
	cv.put(TABLE_NUM, miCount);
	cv.put(TABLE_DATA, data + miCount);
	/* 插入数据 */
	mSQLiteDatabase.insert(TABLE_NAME, null, cv);
	miCount++;
	UpdataAdapter();
}

/* 从表中删除指定的一条数据 */
```



```
            public void DeleteData()
            {
                /* 删除数据 */
                mSQLiteDatabase.execSQL("DELETE FROM " + TABLE_NAME
                                                    + " WHERE _id=" + Integer.toString
(miCount));
                miCount--;
                if (miCount < 0)
                {
                    miCount = 0;
                }
                UpdataAdapter();
            }
            /* 更新试图显示（更新 ListView 数据源适配器） */
            public void UpdataAdapter()
            {
                // 获取数据库 Phones 的 Cursor
                Cursor cur = mSQLiteDatabase.query(TABLE_NAME,
                         new String[] { TABLE_ID, TABLE_NUM, TABLE_DATA }, null, null, null,
null, null);
                miCount = cur.getCount();
                if (cur != null && cur.getCount() >= 0)
                {
                    // ListAdapter 是 ListView 和后台数据的桥梁
                    ListAdapter adapter = new SimpleCursorAdapter(this,
                    // 定义 List 中每一行的显示模板
                        // 表示每一行包含两个数据项
                        android.R.layout.simple_list_item_2,
                        // 数据库的 Cursor 对象
                        cur,
                        // 从数据库的 TABLE_NUM 和 TABLE_DATA 两列中取数据
                        new String[] { TABLE_NUM, TABLE_DATA },
                        // 与 NAME 和 NUMBER 对应的 Views
                        new int[] { android.R.id.text1, android.R.id.text2 });
                    /* 将 adapter 添加到 m_ListView 中 */
                    m_ListView.setAdapter(adapter);
                }
            }
            /* 按键事件处理 */
            public boolean onKeyDown(int keyCode, KeyEvent event)
            {
                if (keyCode == KeyEvent.KEYCODE_BACK)
                {
                    /* 退出时，不要忘记关闭 */
                    mSQLiteDatabase.close();
                    this.finish();
                    return true;
                }
```




```
                return super.onKeyDown(keyCode, event);
            }
        }
```

13.1.4 Content Providers数据存储

Content Providers 是 Android 平台中重要的组件之一，它提供了一套标准的接口用来实现数据的增、删、改、查，可以使各个应用程序之间实现数据共享。我们可以获得系统内部共享资源（如音频、视频、图像和联系人等），还可以自定义应用程序的 Content Providers，为其他应用程序提供访问自身数据的接口。

Content Providers 是如何存储数据的呢？首先 Content Providers 把它的数据作为数据库模型上的单一数据表提供出来，在数据表中，每一行是一条记录，每一列代表着某一特定类型的值。如表 13-1 所示，显示的是联系人信息的数据模型。

表 13-1 Content Providers 数据模型

ID	NUMBER	NUMBER_KEY	LABEL	NAME	TYPE
1	400	400_500	Kirkland office	Anny	TYPE_WORK
2	401	401_500	NY apartment	JAKE	TYPE_MOBILE
3	402	402_500	Kirkland office	Mary	TYPE_HOME
4	403	403_500	Kirkland office	Ingo	TYPE_WORK

每一个 Content Providers 都会对外提供一个 URI 作为其唯一标识自己的数据集（data set），在与其他程序互动的过程中都会使用到这个常量，格式如图 13-17 所示。如果一个 Content Providers 管理多个数据集，其将会为每个数据集分配一个独立的 URI。所有的 Content Providers 的 URI 都以“content：//”开头，其中“content：”是用来标识数据是由 Content Providers 管理的 schema。

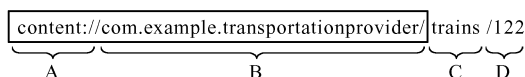


图 13-17 Content Providers URI 组成

那么，Content Providers 如何进行增、删、改、查及自定义 Content Providers 的操作呢，具体实现步骤如下：

（1）创建 Content Providers

创建 Content Providers 需要获取一个 ContentResolver 的实例，可通过 Activity 的成员方法 getContentResovler()方法来获得，代码如下：

```
        ContentResolver cr=getContentResovler() ;
```

ContentResolver 这一方法可实现找到指定的 Content Providers（内容提供者）并获取到 Content Providers 的数据。而在 ContentResolver 查询过程的开始，Android 系统将会确定查询所需具体的 Content Providers，确认它是否启动并运行它。Android 系统还负责初始化所有的 Content Providers，不需要用户自己去创建。实际上，Content Providers 的用户都不可能直接访



问到 Content Providers 实例，只能通过 ContentResolver 从中间进行代理间接实现访问。

（2）查询记录

在 Content Providers 中使用的查询字符串有别于标准的 SQL 查询。诸如 select、add、delete、modify 等操作都使用一种特殊的 URI 来进行，这种 URI 由 3 个部分组成。如“content：//”代表数据的路径，后接一个可选的标识数据的 ID。表 13-2 是一些 URI 示例。

表 13-2　Content Providers URI 示例

示　　例	功　　能
content：//media/internal/images	这个 URI 将返回设备上存储的所有图片
content：//contacts/people/	这个 URI 将返回设备上所有联系人信息
content：//contacts/people/50	这个 URI 返回单个结果（联系人信息中 ID 为 50 的联系人记录）

尽管这种查询字符串格式很常见，但是它看起来还是有些令人迷惑。为此，Android 提供一系列的帮助类（在 android.provider 包下），里面包含了很多以类变量形式给出的查询字符串，这种方式更容易让开发者理解，示例代码如下：

```
MediaStore.Images.Media.INTERNAL_CONTENT_URI;
Contacts.People.CONTENT_URI;
```

因此，如上面所讲的示例 content ://contacts/people/50，这个 URI 就可以写成如下形式：

```
Uri uri = getContentResolver().insert(People.CONTENT_URI, 50);
```

再执行数据查询：

```
Cursor cur = managedQuery(mContacts, columns,null,null, null ) ;
```

这个查询返回一个包含所有数据字段的游标，可以通过迭代这个游标来获取所有的数据。查询功能在 Content Providers 中相当重要，下面通过示例代码巩固这一知识点。代码的主要功能是：依次读取联系人信息表中指定用户的数据列 name 和 number。

```
/**
* 查询
*/
	private void displayDatas() {
		// 该数组中包含了所有要返回的字段
		String columns[] = new String[] { People.NAME, People.NUMBER };
		Uri mContacts = People.CONTENT_URI;
		Cursor cur = managedQuery(mContacts, columns, // 要返回的数据字段
				null, // WHERE 子句
				null, // WHERE 子句的参数
				null // Order-by 子句
		);
		if (cur.moveToFirst()) {
			String name = null;
			String phoneNo = null;
			do {
				// 获取字段的值
```

```
                    name = cur.getString(cur.getColumnIndex(People.NAME));
                    phoneNo = cur.getString(cur.getColumnIndex(People.NUMBER));
                    Toast.makeText(this, name + "" + phoneNo, Toast.LENGTH_LONG)
                            .show();
                } while (cur.moveToNext());
            }
        }
```

（3）修改记录

可以使用 ContentResolver.update()方法来修改数据，具体方法如下：

```
    /**
         * 更改数据
         */
        private void updateRecord(int recNo, String name) {
            Uri uri = ContentUris.withAppendedId(~~People~~.CONTENT_URI, recNo);
            ContentValues values = new ContentValues();
            values.put(~~People~~.NAME, name);
            getContentResolver().update(uri, values, null, null);
        }
```

这样就可以调用上面的方法来更新指定记录，实现代码如下：

```
    updateRecord(12，"XYZ");    //更改第 12 条记录的 name 字段值为“XYZ”
```

（4）添加记录

要增加记录，可以调用 ContentResolver.insert()方法。该方法接收一个要增加的记录的目标 URI 以及一个包含了新记录值的 Map 对象，调用后的返回值是新记录的 URI，包含记录号。

在上面的示例中，都是基于联系人信息簿这个标准的 Content Providers，现在继续用它来创建一个 insertRecord()方法来对联系人信息簿中进行数据的添加：

```
    /**
         * 插入数据
         */
        private void insertRecords(String name, String phoneNo) {
            ContentValues values = new ContentValues();
            values.put(People.NAME, name);
            Uri uri = getContentResolver().insert(People.CONTENT_URI, values);
            Log.d("ANDROID", uri.toString());
            Uri numberUri = Uri.withAppendedPath(uri, People.Phones.CONTENT_DIRECTORY);
            values.clear();
            values.put(Contacts.Phones.TYPE, People.Phones.TYPE_MOBILE);
            values.put(People.NUMBER, phoneNo);
            getContentResolver().insert(numberUri, values);
        }
```

这样就可以调用 insertRecords(name，phoneNo)来向联系人信息簿中添加联系人姓名和电话号码了。

（5）删除记录

Content Providers 中提供了 getContextResolver.delete()方法用来删除记录，下面的代码用



来删除手机设备上所有的联系人信息：

```
/**
 * 删除数据
 */
private void deleteRecords() {
    Uri uri = People.CONTENT_URI;
    getContentResolver().delete(uri, null, null);
}
```

也可以指定 WHERE 条件语句来删除特定的记录，如：

```
//删除 name 为‘ABC’的记录
getContentResolver().delete(uri, "NAME="+ "ABC",null) ;
```

（6）自定义 Content Proviser

可以通过自定义一个继承于 ContentProvider 的类来编写属于自己的 ContentProvider。在该类中通过重写基类中的方法，用来提供给外部操作的接口。如何在方法内部组织和存储数据，则由开发者自定义。数据存储的类型既可以使用数据库存储，也可使用 XML 存储方式。总之，不管采用哪种机制，外部用户关注的仅仅是可操作的接口。在实际的开发中，大多数情况使用的是系统提供的 Content Providers。由于很少情况自定义 Content Providers，而且模式相对较为固定，所以我们在使用时可以参照现有资料或 API 示例即可。

下面通过一个示例讲述 Content Providers 数据存储的使用方法，代码实现功能是读取自建的电话簿信息（ID,TITLE，NOTE）。示例代码运行结果如图 13-18 至图 13-20 所示。



图 13-18　Content Providers 存储示例代码运行结果(1)



图 13-19　Content Providers 存储示例代码运行结果(2)



图 13-20　Content Providers 存储示例代码运行结果(3)

首先，介绍如何通过 xml 布局实现界面效果，如代码清单 13-10 所示。

代码清单 13-10　Content Providers数据存储示例（第 13 章\Demo_13_04）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
```

```
        android:layout_height="fill_parent"
        >
    <TextView
        android:id="@+id/txtView"
        android:layout_width="fill_parent"
        android:layout_height="wrap_content"
        />
    </LinearLayout>
```

其次，介绍 Activity java 代码如何配合布局实现 Content Providers 数据存储操作，如代码清单 13-11 至代码清单 13-14 所示。

代码清单 13-11 Content Providers 数据存储示例（第 13 章\Demo_13_04）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.content.ContentValues;
import android.database.Cursor;
import android.net.Uri;
import android.os.Bundle;
import android.widget.TextView;

/**
 * Content Providers 示例
 * @author
 *
 */
public class MainActivity extends Activity
{
    private TextView textView;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);
        setContentView(R.layout.main);
        //获取控件
        textView=(TextView)findViewById(R.id.txtView);

        /* 插入数据 */
        ContentValues values = new ContentValues();
        values.put(ContactUser.Notes.TITLE, "联系人信息");
        values.put(ContactUser.Notes.NOTE, "我的名字叫 Spring");
        getContentResolver().insert(ContactUser.Notes.CONTENT_URI, values);

        values.clear();
        values.put(ContactUser.Notes.TITLE, "联系人信息");
        values.put(ContactUser.Notes.NOTE, "我的名字叫 Sunning");
        getContentResolver().insert(ContactUser.Notes.CONTENT_URI, values);

        /* 显示 */
        displayNote();
```



```
        }
        /**
         * 显示信息
         */
        private void displayNote()
        {
            String columns[] = new String[] { ContactUser.Notes._ID,
                                               ContactUser.Notes.TITLE,
                                               ContactUser.Notes.NOTE,
                                               ContactUser.Notes.CREATEDDATE,
                                               ContactUser.Notes.MODIFIEDDATE };

            //获取定义的 Uri
            Uri myUri = ContactUser.Notes.CONTENT_URI;
            //获取游标对象
            Cursor cur = managedQuery(myUri, columns, null, null, null);
            //游标开始查询数据
            if (cur.moveToFirst())
            {
                String id = null;
                String title = null;
                String note=null;
                String result="";
                do
                {
                    id = cur.getString(cur.getColumnIndex(ContactUser.Notes._ID));
                    title = cur.getString(cur.getColumnIndex(ContactUser.Notes.TITLE));
    note=cur.getString(cur.getColumnIndex(ContactUser.Notes.NOTE));
                result+="\nID："+id + "\nTITLE：" + title+"\nNOTE：" + note;

                }
                while (cur.moveToNext());
                //显示数据（包括添加的两条数据，还有模拟器中拥有的联系人条数）
                textView.setText(result);
            }
        }

    }
```

代码清单 13-12　Content Providers数据存储示例（第 13 章\Demo_13_04）ContactUserProvider.java

```
    package com.chen.android;
    import java.util.HashMap;
    import android.content.ContentProvider;
    import android.content.ContentUris;
    import android.content.ContentValues;
    import android.content.Context;
    import android.content.UriMatcher;
    import android.content.res.Resources;
    import android.database.Cursor;
```

```
import android.database.SQLException;
import android.database.sqlite.SQLiteDatabase;
import android.database.sqlite.SQLiteOpenHelper;
import android.database.sqlite.SQLiteQueryBuilder;
import android.net.Uri;
import android.text.TextUtils;
import com.chen.android.ContactUser.Notes;
/**
  *  自定义 ContentProvider（实现对数据的增删改查）
  * @author
  *
  */
public class ContactUserProvider extends ContentProvider
{
        //  数据库名
        private static final String         DATABASE_NAME = "contact_user.db";
        private static final int      DATABASE_VERSION     = 2;
        //  表名
        private static final String         NOTES_TABLE_NAME    = "users";
        private static HashMap<String, String>     sNotesProjectionMap;
        private static final int     NOTES= 1;
        private static final int     NOTE_ID= 2;
        private static final UriMatcher        sUriMatcher;
        private DatabaseHelper mOpenHelper;
        //创建表 SQL 语句
        private static final String       CREATE_TABLE="CREATE TABLE "
                                                                + NOTES_TABLE_NAME
                                                                 + " (" + Notes._ID
                                                                 + " INTEGER PRIMARY KEY,"
                                                                 + Notes.TITLE
                                                                 + " TEXT,"
                                                                 + Notes.NOTE
                                                                 + " TEXT,"
                                                                 + Notes.CREATEDDATE
                                                                 + " INTEGER,"
                                                                 + Notes.MODIFIEDDATE
                                                                 + " INTEGER" + ");";

        static
        {
                sUriMatcher = new UriMatcher(UriMatcher.NO_MATCH);
                sUriMatcher.addURI(ContactUser.AUTHORITY, "notes", NOTES);
                sUriMatcher.addURI(ContactUser.AUTHORITY, "notes/#", NOTE_ID);

                sNotesProjectionMap = new HashMap<String, String>();
                sNotesProjectionMap.put(Notes._ID, Notes._ID);
                sNotesProjectionMap.put(Notes.TITLE, Notes.TITLE);
                sNotesProjectionMap.put(Notes.NOTE, Notes.NOTE);
                sNotesProjectionMap.put(Notes.CREATEDDATE, Notes.CREATEDDATE);
                sNotesProjectionMap.put(Notes.MODIFIEDDATE, Notes.MODIFIEDDATE);
        }
```



```
@Override
//ContentProvider 创建时
public boolean onCreate()
{
    mOpenHelper = new DatabaseHelper(getContext());
    return true;
}
@Override
//查询操作
public Cursor query(Uri uri, String[] projection, String selection,
            String[] selectionArgs, String sortOrder)
{
    SQLiteQueryBuilder qb = new SQLiteQueryBuilder();
    switch (sUriMatcher.match(uri))
    {
        case NOTES:
            qb.setTables(NOTES_TABLE_NAME);
            qb.setProjectionMap(sNotesProjectionMap);
            break;

        case NOTE_ID:
            qb.setTables(NOTES_TABLE_NAME);
            qb.setProjectionMap(sNotesProjectionMap);
            qb.appendWhere(Notes._ID + "=" + uri.getPathSegments().get(1));
            break;

        default:
            throw new IllegalArgumentException("Unknown URI " + uri);
    }
    String orderBy;
    if (TextUtils.isEmpty(sortOrder))
    {
        orderBy = ContactUser.Notes.DEFAULT_SORT_ORDER;
    }
    else
    {
        orderBy = sortOrder;
    }
    SQLiteDatabase db = mOpenHelper.getReadableDatabase();
    Cursor c = qb.query(db, projection, selection, selectionArgs, null, null, orderBy);
    c.setNotificationUri(getContext().getContentResolver(), uri);
    return c;
}
@Override
// 如果有自定义类型，必须实现该方法
public String getType(Uri uri)
{
    switch (sUriMatcher.match(uri))
    {
        case NOTES:
            return Notes.CONTENT_TYPE;
```

```
                case NOTE_ID:
                        return Notes.CONTENT_ITEM_TYPE;

                default:
                        throw new IllegalArgumentException("Unknown URI " + uri);
        }
}
@Override
//插入数据库
public Uri insert(Uri uri, ContentValues initialValues)
{
        if (sUriMatcher.match(uri) != NOTES)
        {
                throw new IllegalArgumentException("Unknown URI " + uri);
        }
        ContentValues values;
        if (initialValues != null)
        {
                values = new ContentValues(initialValues);
        }
        else
        {
                values = new ContentValues();
        }
        Long now = Long.valueOf(System.currentTimeMillis());

        if (values.containsKey(ContactUser.Notes.CREATEDDATE) == false)
        {
                values.put(ContactUser.Notes.CREATEDDATE, now);
        }
        if (values.containsKey(ContactUser.Notes.MODIFIEDDATE) == false)
        {
                values.put(ContactUser.Notes.MODIFIEDDATE, now);
        }
        if (values.containsKey(ContactUser.Notes.TITLE) == false)
        {
          Resources r = Resources.getSystem();
          values.put(ContactUser.Notes.TITLE, r.getString(android.R.string.untitled));
        }
        if (values.containsKey(ContactUser.Notes.NOTE) == false)
        {
                values.put(ContactUser.Notes.NOTE, "");
        }
        SQLiteDatabase db = mOpenHelper.getWritableDatabase();
        long rowId = db.insert(NOTES_TABLE_NAME, Notes.NOTE, values);
        if (rowId > 0)
        {
          Uri noteUri = ContentUris.withAppendedId(ContactUser.Notes.CONTENT_URI, rowId);
          getContext().getContentResolver().notifyChange(noteUri, null);
          return noteUri;
        }
```



```
        throw new SQLException("Failed to insert row into " + uri);
}
@Override
//删除数据
public int delete(Uri uri, String where, String[] whereArgs)
{
    SQLiteDatabase db = mOpenHelper.getWritableDatabase();
    int count;
    switch (sUriMatcher.match(uri))
        {
         case NOTES:
                count = db.delete(NOTES_TABLE_NAME, where, whereArgs);
                break;

         case NOTE_ID:
                String noteId = uri.getPathSegments().get(1);
                count = db.delete(NOTES_TABLE_NAME, Notes._ID + "=" + noteId
                + (!TextUtils.isEmpty(where) ? " AND (" + where + ')' : ""), whereArgs);
                break;

          default:
                 throw new IllegalArgumentException("Unknown URI " + uri);
        }
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
}
@Override
//更新数据
public int update(Uri uri, ContentValues values, String where, String[] whereArgs)
{
        SQLiteDatabase db = mOpenHelper.getWritableDatabase();
        int count;
    switch (sUriMatcher.match(uri))
    {
        case NOTES:
                count = db.update(NOTES_TABLE_NAME, values, where, whereArgs);
                break;

        case NOTE_ID:
                String noteId = uri.getPathSegments().get(1);
                count = db.update(NOTES_TABLE_NAME, values, Notes._ID + "="
                            + noteId + (!TextUtils.isEmpty(where) ? " AND (" +
                          where + ')' : ""), whereArgs);
                          break;
        default:
                throw new IllegalArgumentException("Unknown URI " + uri);
}
        getContext().getContentResolver().notifyChange(uri, null);
        return count;
}

/**
```

```
     * 数据库 Helper 类
     */
    private static class DatabaseHelper extends SQLiteOpenHelper
    {
        //构造函数-创建数据库
        DatabaseHelper(Context context)
        {
            super(context, DATABASE_NAME, null, DATABASE_VERSION);
        }
        //创建表
        @Override
        public void onCreate(SQLiteDatabase db)
        {
            db.execSQL(CREATE_TABLE);
        }
        //更新数据库
        @Override
        public void onUpgrade(SQLiteDatabase db, int oldVersion, int newVersion)
        {
            db.execSQL("DROP TABLE IF EXISTS notes");
            onCreate(db);
        }
    }
}
```

代码清单 13-13　Content Providers 数据存储示例（第 13 章\Demo_13_04）ContactUser.java

```
package com.chen.android;

import android.net.Uri;
import android.provider.BaseColumns;
/**
 * 配置联系人类
 * @author
 *
 */
public class ContactUser
{
    //ContentProvider 的 uri
    public static final String    AUTHORITY    = "com.google.provider.NotePad";
    private ContactUser(){}

    // 定义基本字段
    public static final class Notes implements BaseColumns
    {
     private Notes(){}

public static final Uri  CONTENT_URI
                          = Uri.parse("content://" + AUTHORITY + "/notes");

     // 新的 MIME 类型-多个
```



```
            public static final String   CONTENT_TYPE
                                     = "vnd.android.cursor.dir/vnd.google.note";

            // 新的 MIME 类型-单个
            public static final String   CONTENT_ITEM_TYPE
                                     = "vnd.android.cursor.item/vnd.google.note";

        public static final String     DEFAULT_SORT_ORDER   = "modified DESC";

            //字段
            public static final String   TITLE                 = "title";
            public static final String   NOTE                  = "note";
            public static final String   CREATEDDATE           = "created";
            public static final String   MODIFIEDDATE          = "modified";

        }
}
```

代码清单 13-14　Content Providers 数据存储示例（第 13 章\Demo_13_04）ContentProviderDemo.java

```
package com.chen.android;

import android.app.Activity;
import android.database.Cursor;
import android.graphics.Color;
import android.os.Bundle;
import android.provider.Contacts.Phones;
import android.view.View;
import android.widget.AdapterView;
import android.widget.LinearLayout;
import android.widget.ListAdapter;
import android.widget.ListView;
import android.widget.SimpleCursorAdapter;
import android.widget.Toast;

/**
 * ContentProvider 读取电话薄信息
 *
 */
public class ContentProviderDemo extends Activity {
    LinearLayout m_LinearLayout;
    ListView m_ListView;

    /**当 activity 首次创建时响应调用*/
    @Override
    public void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);

        /* 创建 LinearLayout 布局对象 */
        m_LinearLayout = new LinearLayout(this);
        /* 设置布局 LinearLayout 的属性 */
        m_LinearLayout.setOrientation(LinearLayout.VERTICAL);
        m_LinearLayout.setBackgroundColor(android.graphics.Color.BLACK);
```




```
        /* 创建 ListView 对象 */
        m_ListView = new ListView(this);
        LinearLayout.LayoutParams param = new LinearLayout.LayoutParams(
                LinearLayout.LayoutParams.FILL_PARENT,
                LinearLayout.LayoutParams.WRAP_CONTENT);
        m_ListView.setBackgroundColor(Color.BLACK);

        /* 添加 m_ListView 到 m_LinearLayout 布局 */
        m_LinearLayout.addView(m_ListView, param);

        /* 设置显示 m_LinearLayout 布局 */
        setContentView(m_LinearLayout);

        // 获取数据库 Phones 的 Cursor
        Cursor c = getContentResolver().query(Phones.CONTENT_URI, null, null,
                null, null);
        startManagingCursor(c);

        // ListAdapter 是 ListView 和后台数据的桥梁
        ListAdapter adapter = new SimpleCursorAdapter(this,
        // 定义 List 中每一行的显示模板
        // 表示每一行包含两个数据项
                android.R.layout.simple_list_item_2,
                // 数据库的 Cursor 对象
                c,
                // 从数据库的 NAME 和 NUMBER 两列中取数据
                new String[] { Phones.NAME, Phones.NUMBER },
                // 与 NAME 和 NUMBER 对应的 Views
                new int[] { android.R.id.text1, android.R.id.text2 });

        /* 将 adapter 添加到 m_ListView 中 */
        m_ListView.setAdapter(adapter);

        /* 为 m_ListView 视图添加 setOnItemSelectedListener 监听 */
        m_ListView.setOnItemSelectedListener(ItemSelected);
        m_ListView.setOnItemClickListener(OnItemClick);

    }

    /**
     * 实现对 ItemSelected 监听
     */
    ListView.OnItemSelectedListener ItemSelected
                            = new ListView.OnItemSelectedListener() {
        @Override
        public void onItemSelected(AdapterView<?> arg0, View view,
                int position, long id) {
            // TODO Auto-generated method stub
            DisplayToast("滚动到第" + Long.toString(arg0.getSelectedItemId()) + "项");
        }

        @Override
        public void onNothingSelected(AdapterView<?> parent) {
            // TODO Auto-generated method stub
        }
```



```
        };

        /**
         * 实现对 ItemClick 监听
         */
        ListView.OnItemClickListener OnItemClick = new ListView.OnItemClickListener() {

            @Override
            public void onItemClick(AdapterView<?> arg0, View arg1, int arg2,
                    long arg3) {
                // 于对选中的项进行处理
                DisplayToast("选中了第" + Integer.toString(arg2 + 1) + "项");
            }
        };

        /* 显示 Toast */
        public void DisplayToast(String str) {
            Toast.makeText(this, str, Toast.LENGTH_SHORT).show();
        }
    }
```

最后，需要在 AndroidManifest.xml 要添加访问权限，具体如代码清单 13-15。

代码清单 13-15　Content Providers 数据存储示例（第 13 章\Demo_13_04）AndroidManifest.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
  <LinearLayout
    android:orientation="horizontal"
    android:layout_width="fill_parent"
    android:layout_height="wrap_content">
      <Button
        android:id="@+id/btn_GetLocalUserInfo"
        android:text="获取电话薄信息"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"   />
      <Button
        android:id="@+id/btn_DIYUserInfo"
        android:text="自定义 CP 获取数据"
        android:layout_width="wrap_content"
        android:layout_height="wrap_content"   />
  </LinearLayout>

 <TextView
    android:id="@+id/tv_Reuslt"
    android:layout_width="fill_parent"
```

```
        android:layout_height="wrap_content"
        />
</LinearLayout>
```

13.2　专题二：Android多媒体开发

在其他手机平台上实现 VideoPlayer 的开发过程是一个比较具有挑战性的工作，但是在 Android 上实现 VideoPlayer 的开发，显得相对轻松。在 Android 系统中，是通过 MediaPalyer 类实现播放媒体文件的功能（包括视频和音频）。虽然这个类实现起来相对简单，但是还需要控制各种状态，而对于视频文件的播放还需要设置输出界面。

13.2.1　播放音乐

播放音乐涉及 MediaPlayer 类，下面将重点讲解 MediaPlayer 这一个类的常见操作。

（1）如何获得 MediaPlayer 实例

1）可以直接使用 new 的方式，如：

```
MediaPlayer mp = new MediaPlayer();
```

2）可以使用 create 的方式，如：

```
MediaPlayer mp = MediaPlayer.create(this,R.raw.test);//这种设置就不用调用 setDataSource
```

（2）如何设置要播放的文件

MediaPlayer 要播放的文件主要包括 3 个来源：

1）用户在开发过程中，使用事先自带的 resource 资源，例如：MediaPlayer.create(this, R.raw.test);

2）存储在 SD 卡或其他文件路径下的媒体文件，例如：mediaPlayer.setDataSource ("/sdcard/test.mp3");

3）网络上的媒体文件，例如：mediaPlayer.setDataSource("http：//www.citynorth.cn/music /confucius.mp3");

MediaPlayer 的 setDataSource 一共给出了四个方法：

```
setDataSource (String path)
setDataSource (FileDescriptor fd)
setDataSource (Context context， Uri uri)
setDataSource (FileDescriptor fd， long offset， long length)
```

（3）对播放器的主要控制方法

Android 是通过控制播放器的播放状态来控制媒体文件的播放，其中：prepare()和 prepareAsync() 提供了同步和异步两种方式设置播放器进入资源准备状态。需要注意的是，如果 MediaPlayer 实例是由 create 方法创建的，那么第一次启动播放前不需要再调用 prepare()，因为在 create()方法中已经调用过。播放器的主要控制方法总结如表 13-3 所示。



表 13-3　播放器的主要控制方法

播放器的主要控制方法	功 能 说 明
prepare()	提供异步的方式设置播放器进入资源准备状态
prepareAsync()	提供同步的方式设置播放器进入资源准备状态
start()	真正启动文件播放
pause()	播放暂停
stop()	播放停止
seekTo()	可以让播放器从指定的位置开始播放，需要注意的是该方法是个异步方法，也就是说该方法返回时并不意味着定位完成，尤其是播放的网络文件，真正定位完成时会触发 OnSeekComplete.onSeekComplete()方法，如果需要时可以调用 setOnSeekCompleteListener(OnSeekCompleteListener)设置监听器并进行处理
release()	释放播放器占用的资源
reset()	可以使播放器从 Error 状态中恢复过来，重新回到 Idle 状态

（4）设置播放器的监听方法

MediaPlayer 提供了一些设置不同监听器的方法来更好地对播放器的工作状态进行监听，以实现及时处理各种播放情况。例如，提供了 setOnCompletionListener(MediaPlayer.OnCompletionListener listener)方法对播放结束进行监听，一般在播放音频文件结束时调用，在方法中通常加 release()方法释放音频资源；提供了 setOnErrorListener(MediaPlayer.OnErrorListener listener)方法，设置播放出错时的监听方法。

（5）MediaPlayer 的生命周期

MediaPlayer 的生命周期如图 13-21 所示。

这张状态转换图清晰地描述了 MediaPlayer 的各个状态，也列举了主要方法的调用先后过程，每种方法只能在一些特定的状态下使用，如果使用状态不正确则会引发 IllegalStateException 异常。

1）Idle 状态：当使用 new()方法创建一个 MediaPlayer 对象或者调用其 reset()方法时，该 MediaPlayer 对象处于 idle 状态。这两种方法的一个重要差别就是：如果在这个状态下调用了 getDuration()等方法（相当于调用时机不正确）则会通过 reset()方法进入 idle 状态，接着会触发 OnErrorListener.onError()方法，使 MediaPlayer 进入 Error 状态；如果是新创建的 MediaPlayer 对象，则并不会触发 onError()，也不会进入 Error 状态。

2）End 状态：通过 release()方法可以进入 End 状态，只要 MediaPlayer 对象不再被使用，就应当通过 release()方法释放相关的软硬件组件资源。如果 MediaPlayer 对象进入了 End 状态，则不会在进入任何其他状态。

3）Initialized 状态： MediaPlayer 调用 setDataSource()方法时就进入 Initialized 状态，表示此时要播放的文件已经设置好。

4）Prepared 状态：初始化完成之后还需要通过调用 prepare()或 prepareAsync()方法，这两个方法是同步和异步的区别。只有进入 Prepared 状态，才表明 MediaPlayer 到目前为止都没有错误，可以进行文件播放。

5）Preparing 状态：这个状态通常是和 prepareAsync()配合使用，如果异步准备完成，则会触发 OnPreparedListener.onPrepared()，进而进入 Prepared 状态。

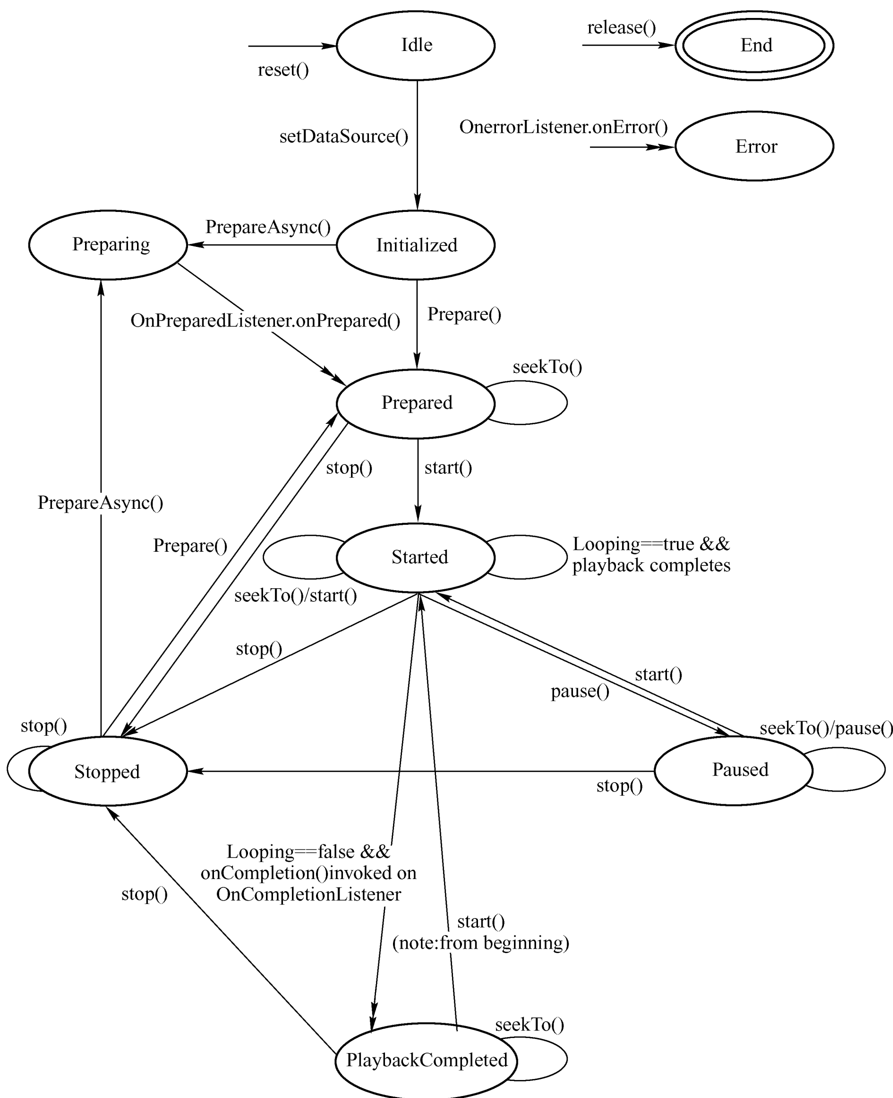


图 13-21　MediaPlayer 的生命周期

6）Started 状态： MediaPlayer 一旦准备好，就可以调用 start()方法，进而 MediaPlayer 就处于 Started 状态，这表明 MediaPlayer 正在播放文件过程中。可以使用 isPlaying()测试 MediaPlayer 是否处于了 Started 状态。如果播放完毕，而又设置了循环播放，则 MediaPlayer 仍然会处于 Started 状态。类似，如果在该状态下 MediaPlayer 调用了 seekTo()或者 start()方法均可以让 MediaPlayer 停留在 Started 状态。

7）Paused 状态：Started 状态下 MediaPlayer 调用 pause()方法可以暂停 MediaPlayer，从而进入 Paused 状态，该状态可以调用 seekTo()方法。MediaPlayer 暂停后再次调用 start()则可以继续进行播放，从而转到 Started 状态。

8）Stop 状态：Started 或者 Paused 状态下均可调用 stop()停止 MediaPlayer，而处于 Stop 状态的 MediaPlayer 要想重新播放，需要通过 prepareAsync()和 prepare()重新回到先前的



Prepared 状态。

9）PlaybackCompleted 状态：文件正常播放完毕，而又没有设置循环播放就会进入该状态，并会触发 OnCompletionListener 的 onCompletion()方法。此时可以调用 start()方法重新播放文件，也可以通过 stop()方法停止 MediaPlayer，还可以通过 seekTo()来重新定位播放的位置。

10）Error 状态：如果由于某种原因 MediaPlayer 出现了错误，会触发 OnErrorListener.onError()事件，此时 MediaPlayer 就进入 Error 状态。及时捕捉并妥善处理这些错误是很重要的，可以帮助我们及时释放相关的软硬件资源，也可以改善用户体验。通过 setOnErrorListener (android.media.MediaPlayer.OnErrorListener)可以设置该监听器。如果 MediaPlayer 进入了 Error 状态，可以通过调用 reset()来恢复，使得 MediaPlayer 重新返回到 Idle 状态。

下面通过一个示例，讲述播放音频的使用方法。如要在模拟器中演示播放音乐的效果，需要在 DDMS 模拟器的 sdcard 存入音频文件，具体步骤如下：

1）运行模拟器，打开 DDMS，在 File Explorer 找到 mnt 文件夹下的 sdcard 文件夹，如图 13-22 所示。

2）单击右上角，导入音频文件，如图 13-23 所示。

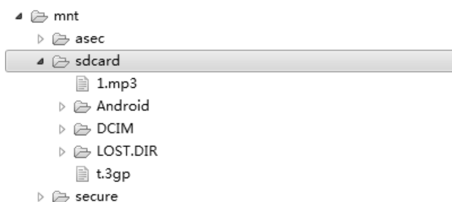


图 13-22　模拟器 DDMS 中 sdcard 的位置



图 13-23　在模拟器中 sdcard 中导入文件

这样，运行示例代码就可以在模拟器上播放音频文件了。示例代码运行结果如图 13-24 所示。



图 13-24　播放音频示例

首先，介绍如何通过 xml 布局实现界面效果，如代码清单 13-16 所示。

代码清单 13-16　播放音频示例（第 13 章\Demo_13_05）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
     android:orientation="vertical"
     android:layout_width="fill_parent"
     android:layout_height="fill_parent">

     <TextView
         android:layout_width="fill_parent"
         android:layout_height="wrap_content"
         android:text="播放音频示例....." />

     <Button
          android:text="播放"
```

```
            android:id="@+id/play"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content">
        </Button>

        <Button
            android:text="暂停"
            android:id="@+id/pause"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content">
        </Button>

        <Button
             android:text="停止"
             android:id="@+id/stop"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content">
        </Button>

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现播放音频功能，如代码清单 13-17 所示。

代码清单 13-17　播放音频示例（第 13 章\Demo_13_05）MainActivity.java

```
package com.chen.android;

import java.io.IOException;
import android.app.Activity;
import android.media.MediaPlayer;
import android.media.MediaPlayer.OnCompletionListener;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;

/**
 * 播放音频（MediaPlayer 的用法）
 *
 * @author
 *
 */
public class MainActivity extends Activity {
        private Button bplay, bpause, bstop;
        private MediaPlayer mp = new MediaPlayer();

        @Override
        public void onCreate(Bundle savedInstanceState) {
                super.onCreate(savedInstanceState);
                /*获取布局对象*/
                setContentView(R.layout.main);
                bplay = (Button) findViewById(R.id.play);
                bpause = (Button) findViewById(R.id.pause);
                bstop = (Button) findViewById(R.id.stop);
              /*设置事件监听*/
```



```
            bplay.setOnClickListener(playMusic);
            bpause.setOnClickListener(pauseMusic);
            bstop.setOnClickListener(stopMusic);
    }

  /**
   * 播放实现
   */
    Button.OnClickListener playMusic = new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                    try {
                            // 就是把音频文件放进 DDMS 中 mnt/sdcard 里面
                            mp.setDataSource("/sdcard/1.mp3");
                            mp.prepare();
                            mp.start();
                    } catch (IllegalArgumentException e) {
                            e.printStackTrace();
                    } catch (IllegalStateException e) {
                            e.printStackTrace();
                    } catch (IOException e) {
                            e.printStackTrace();
                    }
                    //音频播出完毕监听事件
                    mp.setOnCompletionListener(new OnCompletionListener() {
                            @Override
                            public void onCompletion(MediaPlayer mp) {
                                    //释放资源
                                    mp.release();
                            }
                    });
            }
    };

         /**
          *暂停播放
          */
    Button.OnClickListener pauseMusic = new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
                    if (mp != null) {
                            mp.pause();
                    }
            }
    };
       /**
       *停止播放
       */
    Button.OnClickListener stopMusic = new Button.OnClickListener() {
            @Override
            public void onClick(View v) {
```

```
                if (mp != null) {
                    mp.stop();
                }
            }
        };
         /**
          *释放资源
          */
        @Override
        protected void onDestroy() {
            if (mp != null)
                //释放资源
                mp.release();
            super.onDestroy();
        }

    }
```

13.2.2 播放视频

播放视频涉及 VideoView 类，其用法和 MediaPlayer 类似，也有各种播放状态，在这里不赘述，读者可以参考 Android 开发文档。需要注意的是，涉及如何将视频控件显示在屏幕上，需要在界面上方添加一个 VideoView 控件。手机支持的视频格式有两种，分别是.mp4 和.3gp 格式的视频文件。

下面通过一个示例，讲述播放视频的用法，主要包括视频的播放、暂停和停止等。示例代码运行结果如图 13-25 所示。



图 13-25　播放视频示例

首先，介绍如何通过 xml 布局实现界面效果，如代码清单 13-18 所示。

代码清单 13-18　播放视频示例（第 13 章\Demo_13_06）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<AbsoluteLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    >
    <TextView
        android:layout_width="fill_parent"
    android:layout_height="wrap_content"
    android:text="@string/hello"
     />
   <VideoView
    android:id="@+id/VideoView01"
    android:layout_width="320px"
```



```
            android:layout_height="240px"
        />
        <Button android:id="@+id/LoadButton"
        android:layout_width="80px"
        android:layout_height="wrap_content"
        android:text="装载"
            android:layout_x="30px"
            android:layout_y="300px"
        />
        <Button android:id="@+id/PlayButton"
        android:layout_width="80px"
        android:layout_height="wrap_content"
        android:text="播放"
           android:layout_x="120px"
           android:layout_y="300px"
        />
        <Button android:id="@+id/PauseButton"
        android:layout_width="80px"
        android:layout_height="wrap_content"
        android:text="暂停"
           android:layout_x="210px"
          android:layout_y="300px"
         />
</AbsoluteLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现播放视频，如代码清单 13-19 所示。

代码清单 13-19　播放视频示例（第 13 章\Demo_13_06）MainActivity.java

```
package com.chen.android;

import android.app.Activity;
import android.os.Bundle;
import android.view.View;
import android.widget.Button;
import android.widget.MediaController;
import android.widget.VideoView;

/**
 * 播放视频（VideoView 的用法）
 *
 * @author
 *
 */
public class MainActivity extends Activity
{
      VideoView videoView;
    @Override
    public void onCreate(Bundle savedInstanceState)
    {
        super.onCreate(savedInstanceState);

        setContentView(R.layout.main);
```




```
        /* 创建 VideoView 对象 */
         videoView = (VideoView) findViewById(R.id.VideoView01);
        /* 操作播放的三个按钮 */
        Button PauseButton = (Button) this.findViewById(R.id.PauseButton);
        Button LoadButton = (Button) this.findViewById(R.id.LoadButton);
        Button PlayButton = (Button) this.findViewById(R.id.PlayButton);
         /*设置事件监听*/
        LoadButton.setOnClickListener(LoadMusic);
        PlayButton.setOnClickListener(PlayMusic);
        PauseButton.setOnClickListener(PauseMusic);
}
 /**
     * 加载视频文件
     */
        Button.OnClickListener LoadMusic = new Button.OnClickListener() {
                @Override
                public void onClick(View v) {
                        /* 设置路径 */
                        videoView.setVideoPath("/sdcard/t.3gp");
                        /* 设置模式-播放进度条 */
                        videoView.setMediaController(new MediaController(MainActivity.this));
                        videoView.requestFocus();
                }
        };
            /**
             *播放视频
             */
        Button.OnClickListener PlayMusic = new Button.OnClickListener() {
                @Override
                public void onClick(View v) {
                        /* 开始播放 */
                        if(videoView!=null)
                        {
                        videoView.start();
                        }
                }
        };
         /**
         *暂停播放
         */
        Button.OnClickListener PauseMusic = new Button.OnClickListener() {
                @Override
                public void onClick(View v) {
                        if (videoView != null) {
                                /* 暂停 */
                                videoView.pause();
                        }
```



```
            }
        };
    }
```

13.3　专题三：外部接口编程

因为订票系统的查询功能实现起来不是太容易，因此在本书中曾多次强调可以使用 Web 第三方接口（简称 Web 服务）实现机票相关功能。什么是 Web 服务呢？通俗的理解那就是：在应用程序开发端通过 HTTP 通信，调用其他公司发布的功能接口实现后台操作，调用 Web 服务关注的是前端手机开发的实现，无需关注某些功能后台的实现。Web 服务比较专业的定义是：由企业发布的完成某一特定商务需求的在线应用服务，而其他公司或应用软件可以通过 Internet 来访问并使用这项在线服务。它主要是基于 XML 和 HTTPS 的一种服务，其通信协议主要基于 SOAP，服务的描述通过 WSDL，通过 UDDI 来发现和获得服务的元数据。在几年前 Web 服务曾风靡一时，现今也还广泛使用，特别是开发一些共同服务平台时，需要调用 Web 接口。例如，要想开发一个翻译软件或者天气预报软件，可以调用 Google 的 API 翻译接口或者天气预报接口，这样会使我们的开发过程事半功倍，提供更为丰富的应用程序服务。

下面通过一个示例讲述如何调用第三方接口的过程，信息的来源由 Google 提供的翻译和天气预报信息。主要功能是：输入中文城市名称，显示近三天该城市的天气预报信息，示例代码运行结果如图 13-26 所示。



图 13-26　外部接口（城市天气预报）使用示例代码运行结果

首先，介绍如何通过 xml 布局实现界面效果，如代码清单 13-20 所示。

代码清单 13-20　外部接口的使用示例（第 13 章\CityWeather）main.xml

```
<?xml version="1.0" encoding="utf-8"?>
<LinearLayout
    xmlns:android="http://schemas.android.com/apk/res/android"
    android:orientation="vertical"
    android:layout_width="fill_parent"
    android:layout_height="fill_parent"
    android:background="@drawable/bg"
    >
    <!-- 第一行字 -->
     <TextView
            android:id="@+id/TextView001"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"
              android:text="请输入城市的名字(中文)："
              android:textStyle="bold"
              android:textSize="16px"
```




```
            android:layout_marginLeft="10px"
            android:textColor = "#ff8c00"
            >
    </TextView>
     <!-- 第二行 -->
    <TableLayout
        android:id="@+id/TableLayout002"
         android:layout_height="wrap_content"
         android:layout_width="fill_parent">
          <TableRow
              android:id="@+id/TableRow001"
              android:layout_height="wrap_content"
              android:layout_width="fill_parent">
                      <TextView
                      android:id="@+id/TextView002"
                      android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:text="城市 :"
                      android:textStyle="bold"
                      android:textSize="16px"
                      android:layout_marginLeft="10px"
                      android:textColor = "#ff8c00"
                      >
                      </TextView>
                      <EditText
                        android:id="@+id/ed_words"
                        android:layout_height="wrap_content"
                        android:layout_width="fill_parent"
                        android:paddingLeft="10px"
                        android:minWidth="200px">
                        </EditText>
                      <Button
                      android:id="@+id/btn_search"
                      android:layout_width="wrap_content"
                      android:layout_height="wrap_content"
                      android:text="确定"
                      android:paddingLeft="10px"></Button>
                </TableRow>
    </TableLayout>
     <!-- 第三行 -->
    <TableLayout
          android:id="@+id/TableLayout01"
          android:layout_width="fill_parent"
          android:layout_height="wrap_content">
                <TableRow
                     android:id="@+id/TableRow02"
                      android:layout_width="wrap_content"
                      android:layout_height="wrap_content">
                        <com.chenchuncha.weather.SingleWeatherInfoView
                            android:id="@+id/weather_0"
```



```
                android:layout_width="wrap_content"
                android:layout_height="wrap_content"/>
        </TableRow>
        <TableRow
          android:id="@+id/TableRow03"
          android:layout_width="wrap_content"
          android:layout_height="wrap_content">
             <com.chenchuncha.weather.SingleWeatherInfoView
             android:id="@+id/weather_1"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"/>
        </TableRow>
        <TableRow
            android:id="@+id/TableRow04"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
             <com.chenchuncha.weather.SingleWeatherInfoView
             android:id="@+id/weather_2"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"/>
        </TableRow>
        <TableRow
            android:id="@+id/TableRow05"
            android:layout_width="wrap_content"
            android:layout_height="wrap_content">
             <com.chenchuncha.weather.SingleWeatherInfoView
             android:id="@+id/weather_3"
             android:layout_width="wrap_content"
             android:layout_height="wrap_content"/>
        </TableRow>
        <TableRow android:id="@+id/TableRow06"
           android:layout_width="wrap_content"
           android:layout_height="wrap_content">
             <com.chenchuncha.weather.SingleWeatherInfoView
              android:id="@+id/weather_4"
              android:layout_width="wrap_content"
              android:layout_height="wrap_content"/>
        </TableRow>
    </TableLayout>

</LinearLayout>
```

其次，介绍 Activity　java 代码如何配合布局实现外部接口，如代码清单 13-21 至代码清单 13-28 所示。

代码清单 13-21　外部接口的使用示例（第 13 章\CityWeather）MainActivity.java

```
package com.chenchuncha.weather;

import java.io.InputStreamReader;
import java.io.UnsupportedEncodingException;
import java.net.MalformedURLException;
```




```
import java.net.URL;
import java.net.URLEncoder;
import javax.xml.parsers.SAXParser;
import javax.xml.parsers.SAXParserFactory;
import org.xml.sax.InputSource;
import org.xml.sax.XMLReader;
import android.app.Activity;
import android.os.Bundle;
import android.util.Log;
import android.view.Gravity;
import android.view.View;
import android.view.View.OnClickListener;
import android.widget.Button;
import android.widget.EditText;
import android.widget.Toast;
/**
 * 程序入口
 *
 */
public class MainActivity extends Activity {
	// 查天气的外部接口
	private String queryString_intput = "http://www.google.com/ig/api?hl=zh_cn&weather=";
	// 翻译软件外部接口
	private String translatorUrl = "http://ajax.googleapis.com/ajax/services/language/translate?"
			+ "v=1.0&langpair=zh-CN|en&";

	private final String DEBUG_TAG = "MainActivity";

	@Override
	public void onCreate(Bundle savedInstanceState) {
		super.onCreate(savedInstanceState);
		setContentView(R.layout.main);

		Button submit_input = (Button) findViewById(R.id.btn_search);
		submit_input.setOnClickListener(new OnClickListener() {
			@Override
			public void onClick(View v) {
				EditText inputcity = (EditText) findViewById(R.id.ed_words);
				// 读取控件文本框信息
				String tmp = inputcity.getText().toString();
				if (tmp.length() <= 0) {
					DisplayToast("请输入城市！");

				} else {
					// 将中文城市翻译成英文
					tmp = TransToEnglish(tmp);

					try {
						URL url = new URL(queryString_intput + tmp);
						// 读取城市天气预报信息
						getCityWeather(url);
					} catch (Exception e) {
						Log.e("CityWeather", e.toString());
```



```
                        }
                    }
                }

            });
        }

        /**
         * 调用接口进行翻译
         *
         * @param tmp
         * @return
         */
        private String TransToEnglish(String tmp) {
            String params = null;

            try {
                // 传递的参数设定编码格式，以防止乱码
                params = URLEncoder.encode("q", "UTF-8") + "="
                        + URLEncoder.encode(tmp, "UTF-8");
            } catch (UnsupportedEncodingException e) {
                Log.e(DEBUG_TAG, "注册方法中参数传递错误！");
            }
            // 得到字符串集合
            String transStr = null;
            transStr = ConnUrlHelper.getPostHttpURLConnByUrl(translatorUrl, params);

            int textIndex = transStr.indexOf("translatedText") + 17;
            int textLen = transStr.indexOf("\"", textIndex) - textIndex;
            transStr = transStr.substring(textIndex, textIndex + textLen);

            return transStr;
        }

        // 更新显示实时天气信息
        private void updateWeatherInfoView(int aResourceID,
                WeatherCurrentCondition aWCC) throws MalformedURLException {

            URL imgURL = new URL("http://www.google.com/" + aWCC.getIcon());
            ((SingleWeatherInfoView) findViewById(aResourceID))
                    .setWeatherIcon(imgURL);
            ((SingleWeatherInfoView) findViewById(aResourceID))
                    .setWeatherString(aWCC.toString());
        }

        // 更新显示天气预报
        private void updateWeatherInfoView(int aResourceID,
                WeatherForecastCondition aWFC) throws MalformedURLException {

            URL imgURL = new URL("http://www.google.com/" + aWFC.getIcon());
            ((SingleWeatherInfoView) findViewById(aResourceID))
                    .setWeatherIcon(imgURL);
            ((SingleWeatherInfoView) findViewById(aResourceID))
                    .setWeatherString(aWFC.toString());
        }
```

```
        // 获取天气信息
        // 通过网络获取数据
        // 传递给 XMLReader 解析
        public void getCityWeather(URL url) {
                try {
                        SAXParserFactory spf = SAXParserFactory.newInstance();
                        SAXParser sp = spf.newSAXParser();

                        XMLReader xr = sp.getXMLReader();

                        GoogleWeatherHandler gwh = new GoogleWeatherHandler();
                        xr.setContentHandler(gwh);

                        InputStreamReader isr = new InputStreamReader(url.openStream(),
                                        "GBK");
                        InputSource is = new InputSource(isr);

                        xr.parse(is);

                        WeatherControl ws = gwh.getMyWeatherSet();

                        // 更新天气信息数据
                        updateWeatherInfoView(R.id.weather_0, ws.getMyCurrentCondition());
                        updateWeatherInfoView(R.id.weather_1, ws.getMyForecastConditions()
                                        .get(0));
                        updateWeatherInfoView(R.id.weather_2, ws.getMyForecastConditions()
                                        .get(1));
                        updateWeatherInfoView(R.id.weather_3, ws.getMyForecastConditions()
                                        .get(2));
                        updateWeatherInfoView(R.id.weather_4, ws.getMyForecastConditions()
                                        .get(3));
                } catch (Exception e) {
                        Log.e("CityWeather", e.toString());
                }
        }

        /* 显示 Toast */
        public void DisplayToast(String str) {
                Toast toast = Toast.makeText(this, str, Toast.LENGTH_SHORT);
                // 设置 toast 显示的位置
                toast.setGravity(Gravity.TOP, 0, 220);
                // 显示该 toast
                toast.show();
        }
}
```

代码清单 13-22　外部接口的使用示例（第 13 章\CityWeather）ConnUrlHelper.java

```
package com.chenchuncha.weather;

import java.io.BufferedReader;
import java.io.DataOutputStream;
import java.io.IOException;
```



```
import java.io.InputStreamReader;
import java.net.HttpURLConnection;
import java.net.MalformedURLException;
import java.net.URL;
import android.util.Log;

/**
 * 数据通信类
 * @author
 *
 */
public abstract class ConnUrlHelper {
    private static String DEBUG_TAG = "ConnUrlHelper";

    /**
     * 辅助方法，用于把流转换为字符串
     */
    public static String convertStreamToString1(InputStreamReader is) {
        BufferedReader reader = new BufferedReader(is);
        StringBuilder sb = new StringBuilder();

        String line = null;
        try {
            while ((line = reader.readLine()) != null) {
                sb.append(line + "\n");
            }
        } catch (IOException e) {
            Log.e(DEBUG_TAG, "the convertStreamToString1 methods IOException");
        } finally {
            try {
                is.close();
            } catch (IOException e) {
                Log.e(DEBUG_TAG,
                        "the convertStreamToString1 methods IOException");
            }
        }
        return sb.toString();
    }

    /**
     * 通过 POST 方式进行通信
     *
     * @param url1
     * @return
     */
    public static String getPostHttpURLConnByUrl(String httpUrl, String params) {
        // 获得的数据
        String resultData = "";
        URL url = null;
        try {
            // 构造一个 URL 对象
            url = new URL(httpUrl);
```




```
        } catch (MalformedURLException e) {
            Log.e(DEBUG_TAG,
                    "the getPostHttpURLConnByUrl methods  MalformedURLException erro");

        }
        if (url != null) {
            try {
            // 使用 HttpURLConnection 打开连接
            HttpURLConnection urlConn = (HttpURLConnection) url
                        .openConnection();
            // 因为这个是 POST 请求，设立需要设置为 true
            urlConn.setDoOutput(true);
            // 设置以 POST 方式
            urlConn.setRequestMethod("POST");
            // POST 请求不能使用缓存
            urlConn.setUseCaches(false);
            // URLConnection.setInstanceFollowRedirects 是成员函数，仅作用于当前函数
            urlConn.setInstanceFollowRedirects(true);
            /* 配置本次连接的 Content-type，
             *配置为 application/x-www-form-urlencoded 的 urlencoded 编码过的 form 参数
             */
             urlConn.setRequestProperty("Content-Type",
                        "application/x-www-form-urlencoded");
             /*取得连接*/
             urlConn.connect();
            // DataOutputStream 流
            DataOutputStream out = new DataOutputStream(
                        urlConn.getOutputStream());
            // 将要上传的内容写入流中
            out.writeBytes(params);
            // 刷新、关闭
            out.flush();
            out.close();
            // 获取数据
            InputStreamReader in = null;
            in = new InputStreamReader(urlConn.getInputStream());
            resultData = convertStreamToString1(in);
            // 关闭 InputStreamReader
            in.close();
            // 关闭 http 连接
            urlConn.disconnect();
            } catch (IOException e) {
                // 查看输出异常
                Log.e(DEBUG_TAG,
                        "the getPostHttpURLConnByUrl methods  IOException erro");
            }
        } else {
            Log.e(DEBUG_TAG,
                    "the getPostHttpURLConnByUrl methods  Url NULL erro");
        }
        return resultData;
```



```
        }

    }
```

代码清单 13-23　外部接口的使用示例（第 13 章\CityWeather）GoogleWeatherHandler.java

```
package com.chenchuncha.weather;
import org.xml.sax.Attributes;
import org.xml.sax.SAXException;
import org.xml.sax.helpers.DefaultHandler;
import android.util.Log;
/**
 * 自定义 handler 类
 * 继承 DefaultHandler
 *
 */
public class GoogleWeatherHandler extends DefaultHandler
{
    //天气信息
    private WeatherControl          myWeatherSet = null;
    //实时天气信息
    private boolean             is_Current_Conditions  = false;
    //预报天气信息
    private boolean             is_Forecast_Conditions = false;
    //标记两个节点
    private final String    CURRENT_CONDITIONS          = "current_conditions";
    private final String    FORECAST_CONDITIONS         = "forecast_conditions";

    public GoogleWeatherHandler()
    {

    }

    //返回天气信息对象
    public WeatherControl getMyWeatherSet()
    {
        return myWeatherSet;
    }

    /**
     * 回调函数用于文档结束
     * 功能：用于处理文档解析的结束事件
     */
    @Override
    public void endDocument() throws SAXException
    {
        super.endDocument();
    }

    /*
    * 处理元素结束事件
    *     uri ：元素的命名空间
```




```
*       localName  ：  元素的本地名称（不带前缀）
*       name  ：  元素的限定名（带前缀）
*/
@Override
public void endElement(String uri, String localName, String name) throws SAXException
{
        if (localName.equals(CURRENT_CONDITIONS))
        {
                this.is_Current_Conditions = false;
        }
        else if (localName.equals(FORECAST_CONDITIONS))
        {
                this.is_Forecast_Conditions = false;
        }
}

/**
  * 回调函数用于文档开始
  * 功能：用于处理文档解析开始时间
  */
@Override
public void startDocument() throws SAXException
{
        this.myWeatherSet = new WeatherControl();
}

  /*
*  处理元素开始事件
* uri：元素的命名空间
* localName  ：  元素的本地名称（不带前缀，节点名）
* qName  ：  元素的限定名（带前缀）
* attributes  ：  元素的属性集合，节点属性
*  功能：    从参数中可以获取元素所在空间 URL，元素名称，属性列表等信息
*/
@Override
public void startElement(String uri, String localName, String qName, Attributes attributes) throws
SAXException
{
        if (localName.equals(CURRENT_CONDITIONS))
        {//实时天气
                Log.i("localName+CURRENT", localName);
                this.myWeatherSet.setMyCurrentCondition(new WeatherCurrentCondition());
                Log.i("localName+CURRENT+1", localName);
                this.is_Current_Conditions = true;
        }
        else if (localName.equals(FORECAST_CONDITIONS))
        {//预报天气
                this.myWeatherSet.getMyForecastConditions().add(new WeatherForecastCondition());
                this.is_Forecast_Conditions = true;
        }
        else
```



```
{
	//分别将得到的信息设置到指定的对象中
	if (localName.equals(CURRENT_CONDITIONS))
	{
		Log.i("localName+CURRENT", localName);
	}
	//得到某一节点的属性值
	String dataAttribute = attributes.getValue("data");

	if (localName.equals("icon"))
	{
		//若是实时天气
		if (this.is_Current_Conditions)
		{
		//赋值
	this.myWeatherSet.getMyCurrentCondition().setIcon(dataAttribute);
		}//若是预测天气
		else if (this.is_Forecast_Conditions)
		{
			this.myWeatherSet.getLastForecastCondition().setIcon(dataAttribute);
		}
	}
	else if (localName.equals("condition"))
	{
		if (this.is_Current_Conditions)
		{
			this.myWeatherSet.getMyCurrentCondition().setCondition(dataAttribute);
		}
		else if (this.is_Forecast_Conditions)
		{
			this.myWeatherSet.getLastForecastCondition().setCondition(dataAttribute);
		}
	}
	else if (localName.equals("temp_c"))
	{
		this.myWeatherSet.getMyCurrentCondition().setTemp_celcius(dataAttribute);
	}
	else if (localName.equals("temp_f"))
	{
		this.myWeatherSet.getMyCurrentCondition().setTemp_fahrenheit(dataAttribute);
	}
	else if (localName.equals("humidity"))
	{
		this.myWeatherSet.getMyCurrentCondition().setHumidity(dataAttribute);
	}
	else if (localName.equals("wind_condition"))
	{
		this.myWeatherSet.getMyCurrentCondition().setWind_condition(dataAttribute);
	}//标题为天气预报条件
	else if (localName.equals("day_of_week"))
```




```
                    {
                        this.myWeatherSet.getLastForecastCondition().setDay_of_week(dataAttribute);
                    }
                    else if (localName.equals("low"))
                    {
                        this.myWeatherSet.getLastForecastCondition().setLow(dataAttribute);
                    }
                    else if (localName.equals("high"))
                    {
                        this.myWeatherSet.getLastForecastCondition().setHigh(dataAttribute);
                    }
                }
            }
            /**
             * 接收字符数据的事件
             */
            @Override
            public void characters(char ch[], int start, int length)
            {
            }
        }
```

代码清单 13-24　外部接口的使用示例（第 13 章\CityWeather）WeatherControl.java

```
package com.chenchuncha.weather;

import java.util.ArrayList;
/**
 *天气信息逻辑处理类
 *
 */
public class WeatherControl
{
    //实时天气信息
    private WeatherCurrentCondition myCurrentCondition = null;

    //预报后四天的天气信息
    private ArrayList<WeatherForecastCondition> myForecastConditions =
                                        new ArrayList<WeatherForecastCondition>();

    public WeatherControl()
    {

    }

    //得到实时天气信息的对象
    public WeatherCurrentCondition getMyCurrentCondition()
    {
        return myCurrentCondition;
    }

    //设置实时天气信息的对象
    public void setMyCurrentCondition(WeatherCurrentCondition myCurrentCondition)
```



```
		{
			this.myCurrentCondition = myCurrentCondition;
		}
		//得到预报天气
		public ArrayList<WeatherForecastCondition> getMyForecastConditions()
		{
			return myForecastConditions;
		}
		/*得到最后一个预报天气
		这里我们每次添加一个数据都是在最后
		所以得到最后一个*/
		public WeatherForecastCondition getLastForecastCondition()
		{
			return myForecastConditions.get(myForecastConditions.size() - 1);
		}
	}
```

代码清单 13-25　外部接口的使用示例（第 13 章\CityWeather）WeatherSet.java

```
	package com.chenchuncha.weather;
	import java.util.ArrayList;
	/**
	 * 天气预报处理类
	 * @author
	 *
	 */
	public class WeatherSet
	{
		//实时天气信息
		private WeatherCurrentCondition myCurrentCondition = null;
		//预报后四天的天气信息
		private ArrayList<WeatherForecastCondition> myForecastConditions =
								new ArrayList<WeatherForecastCondition>();

		public WeatherSet()
		{

		}

		//得到实时天气信息的对象
		public WeatherCurrentCondition getMyCurrentCondition()
		{
			return myCurrentCondition;
		}
		//设置实时天气信息的对象
		public void setMyCurrentCondition(WeatherCurrentCondition myCurrentCondition)
		{
			this.myCurrentCondition = myCurrentCondition;
		}
```

```
        //得到预报天气
        public ArrayList<WeatherForecastCondition> getMyForecastConditions()
        {
                return myForecastConditions;
        }
        /*得到最后一个预报天气
        这里我们每次添加一个数据都是在最后
        所以得到最后一个*/
        public WeatherForecastCondition getLastForecastCondition()
        {
                return myForecastConditions.get(myForecastConditions.size() - 1);
        }
}
```

代码清单 13-26 外部接口的使用示例（第 13 章\CityWeather）WeatherCurrentCondition.java

```
package com.chenchuncha.weather;
import java.io.BufferedInputStream;
import java.io.InputStream;
import java.net.URL;
import java.net.URLConnection;
import android.content.Context;
import android.graphics.Bitmap;
import android.graphics.BitmapFactory;
import android.graphics.Color;
import android.util.AttributeSet;
import android.widget.ImageView;
import android.widget.LinearLayout;
import android.widget.TextView;
/**
 * 每一天的天气预报信息子项
 * @author
 *
 */
public class SingleWeatherInfoView extends LinearLayout
{
        private ImageView      myWeatherImageView = null;
        private TextView       myTempTextView = null;
        public SingleWeatherInfoView(Context context)
        {
                super(context);
        }
    /**
     * 构造每一条天气信息子项
     * @param context
     * @param attrs
     */
```



```
		public SingleWeatherInfoView(Context context, AttributeSet attrs)
		{
			super(context, attrs);

			this.myWeatherImageView = new ImageView(context);
			this.myWeatherImageView.setPadding(10, 5, 5, 5);

			this.myTempTextView = new TextView(context);
			this.myTempTextView.setTextColor(Color.BLACK);
			this.myTempTextView.setTextSize(16);

			//在 LinearLayout 中增加子控件
			this.addView(this.myWeatherImageView,
					new
LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.WRAP_CONTENT,
					android.view.ViewGroup.LayoutParams.WRAP_CONTENT));
			this.addView(this.myTempTextView,
						new  LinearLayout.LayoutParams(android.view.ViewGroup.LayoutParams.
WRAP_CONTENT,
					android.view.ViewGroup.LayoutParams.WRAP_CONTENT));
		}

	/**
	 *  设置天气信息字符串
	 * @param aWeatherString
	 */
		public void setWeatherString(String aWeatherString)
		{
			this.myTempTextView.setText(aWeatherString);
		}

	 /**
	  *  下载天气信息图标
	  * @param aURL
	  */
		public void setWeatherIcon(URL aURL){
			try{
				URLConnection conn = aURL.openConnection();
				conn.connect();
				InputStream is = conn.getInputStream();
				BufferedInputStream bis = new BufferedInputStream(is);
				//将信息流转换成图标
				Bitmap bm = BitmapFactory.decodeStream(bis);
				bis.close();
				is.close();
				this.myWeatherImageView.setImageBitmap(bm);
			}catch (Exception e){}
		}
	}
```

代码清单 13-27　外部接口的使用示例（第 13 章\CityWeather）WeatherCurrentCondition.java

```
	package com.chenchuncha.weather;
```

```
/**
 * 实时天气信息处理(当前天气信息)
 */
public class WeatherCurrentCondition
{
    private String  condition;          // 多云
    private String  temp_celcius;       // 摄氏温度
    private String  temp_fahrenheit;    // 华氏温度
    private String  humidity;           // 湿度:58%
    private String  wind_condition;     // 风向...
    private String  icon;               // 图标

    public WeatherCurrentCondition()
    {

    }
    //得到 Condition（多云）
    public String getCondition()
    {
        return condition;
    }
    //设置 Condition（多云）
    public void setCondition(String condition)
    {
        this.condition = condition;
    }
    //得到设置温度
    public String getTemp_c()
    {
        return temp_celcius;
    }
    //得到华氏温度
    public String getTemp_f()
    {
        return temp_fahrenheit;
    }
    //设置摄氏温度
    public void setTemp_celcius(String temp_celcius)
    {
        this.temp_celcius = temp_celcius;
    }
    //设置华氏温度
    public void setTemp_fahrenheit(String temp_fahrenheit)
    {
        this.temp_fahrenheit = temp_fahrenheit;
    }
```



```
        //得到（湿度:58%）
        public String getHumidity()
        {
            return humidity;
        }
        //设置（湿度:58%）
        public void setHumidity(String humidity)
        {
            this.humidity = humidity;
        }
        //得到风向指示
        public String getWind_condition()
        {
            return wind_condition;
        }
        //设置风向指示
        public void setWind_condition(String wind_condition)
        {
            this.wind_condition = wind_condition;
        }
        //得到图标地址
        public String getIcon()
        {
            return icon;
        }
        //设置图标地址
        public void setIcon(String icon)
        {
            this.icon = icon;
        }
        //得到一个封装打包的字符串，包括除 icno 外的所有东西
        @Override
        public String toString()
        {
            StringBuilder sb = new StringBuilder();
            sb.append("实时天气: ").append(temp_celcius).append(" °C");
            sb.append(" ").append(temp_fahrenheit).append(" F");
            sb.append(" ").append(condition);
            sb.append(" ").append(humidity);
            sb.append(" ").append(wind_condition);
            return sb.toString();
        }
    }
```

代码清单 13-28　外部接口的使用示例（第 13 章\CityWeather）WeatherForecastCondition.java

```
    package com.chenchuncha.weather;
```




```
/**
 * 预报后三天的天气信息
 */
public class WeatherForecastCondition {
    private String day_of_week;        //星期
    private String low;                //最低温度
    private String high;               //最高温度
    private String icon;               //图标
    private String condition;          //提示

    public WeatherForecastCondition()
    {

    }

    public String getCondition()
    {
        return condition;
    }

    public void setCondition(String condition)
    {
        this.condition = condition;
    }

    public String getDay_of_week()
    {
        return day_of_week;
    }

    public void setDay_of_week(String day_of_week)
    {
        this.day_of_week = day_of_week;
    }

    public String getLow()
    {
        return low;
    }

    public void setLow(String low)
    {
        this.low = low;
    }

    public String getHigh()
    {
        return high;
    }
```



```
        public void setHigh(String high)
        {
            this.high = high;
        }

        public String getIcon()
        {
            return icon;
        }

        public void setIcon(String icon)
        {
            this.icon = icon;
        }

        @Override
        public String toString()
        {
            //周日：30/18℃以晴为主
            StringBuilder sb = new StringBuilder();

            sb.append(" ").append(day_of_week);
            sb.append(" : ").append(high);
            sb.append("/").append(low).append(" °C");
            sb.append(" ").append(condition);
            return sb.toString();
        }
    }
```

附　　录

附录A　IIS部署及网站发布

在该附录中介绍了 Windows XP 版和 Windows 7 版 IIS 部署及网站发布，若仅用于测试，不发布到公网上，其中“访问用户的权限设置”、“入站规则设置”是非必需的配置项。

1．Windows XP版

（1）安装与部署 **IIS**

若使用的是 Windows XP 操作系统，系统中还未安装 IIS 服务器，可单击“开始”→“设置”→“控制面板”→“添加/删除程序”，在弹出的对话框中选择“添加/删除 Windows 组件”，在 Windows 组件向导对话框中选中“Internet 信息服务（IIS）”，如图 A−1 所示。然后单击“下一步”，按向导指示，完成对 IIS 的安装。如果在安装 IIS 过程中提示系统未找到 i386（可能是电脑操作系统在安装过程中未安装此文件造成的）则需要到网上下载 i386 文件，然后再重新点击安装 IIS。



图 A−1　安装 IIS

（2）启动 IIS

Internet 信息服务简称为 IIS，单击 Windows“开始”→“设置”→“控制面板”→“Internet 信息服务（IIS）”，即可启动“Internet 信息服务”管理工具。

（3）部署网站及权限设置

打开“Internet 信息服务（IIS）”，右击“默认网站”，选择“新建”→“虚拟目录”，如图 A−2 所示。然后单击“下一步”，输入别名（如 GanaSky）如图 A−3 所示。单击“下一步”，找到项目所在的位置（如图 A−4 所示）。单击“下一步”，选择访问权限（如图 A−5 所示）完成配置。





图 A-2 在 IIS 中部署项目（一）

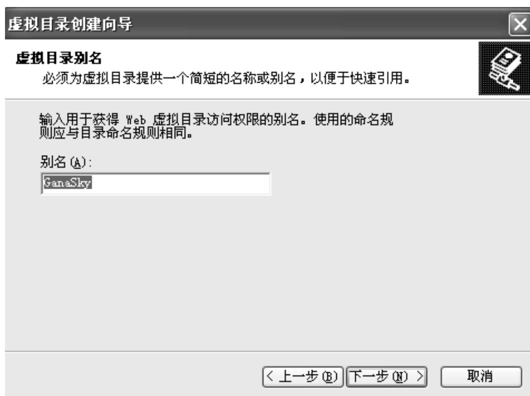


图 A-3 在 IIS 中部署项目（二）



图 A-4 在 IIS 中部署项目（三）

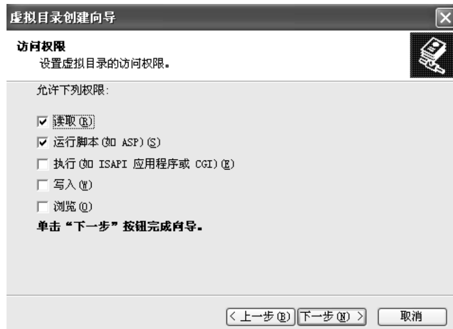


图 A-5 在 IIS 中部署项目（四）

（4）访问用户的权限设置（非必须）

确保程序及资源文件有用户访问权限，如果没有，右击要发布的程序（如 GanaSky），选择“编辑权限”，跳转到“安全”选项卡，为程序添加 Everyone 用户，且 Everyone 的访问权限至少是只读，还可以是读写权限，如图 A-6 至 A-10 所示：



图 A-6 访问用户的权限设置（一）

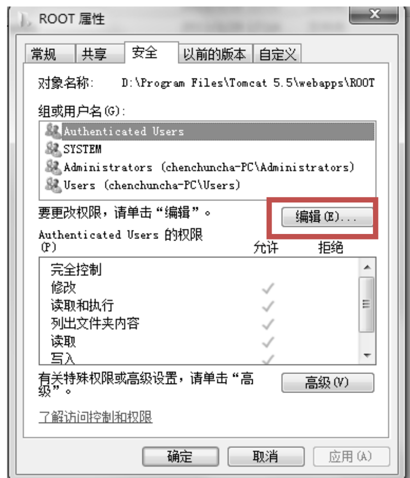


图 A-7 访问用户的权限设置（二）

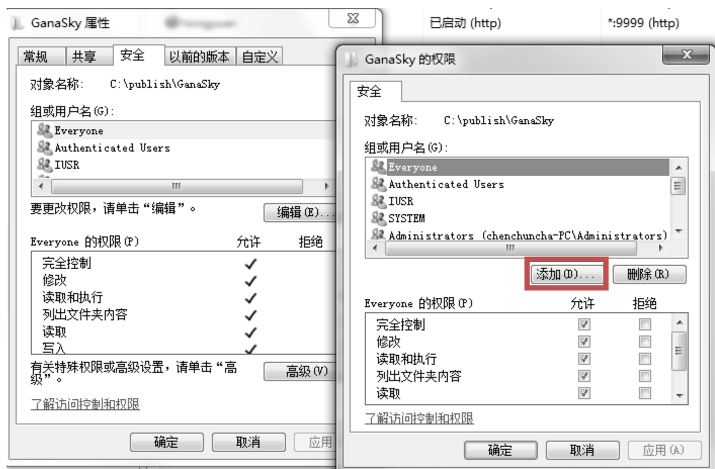


图 A-8　访问用户的权限设置（三）

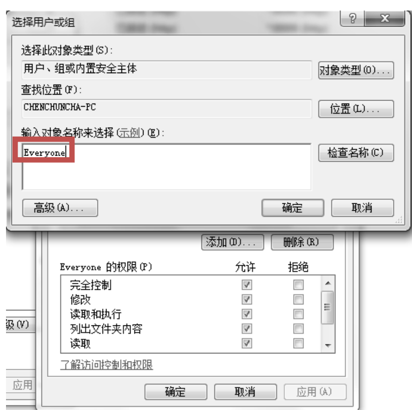


图 A-9　访问用户的权限设置（四）

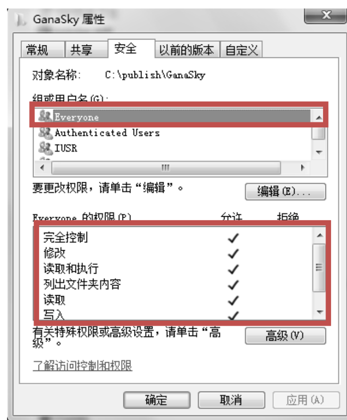


图 A-10　访问用户的权限设置（五）

（5）入站规则设置（非必须）

在高级防火墙中设置入账规则：单击“开始”→“程序”→“控制面板”→“高级防火墙”→“定义入站规则”。如图 A-11 所示。

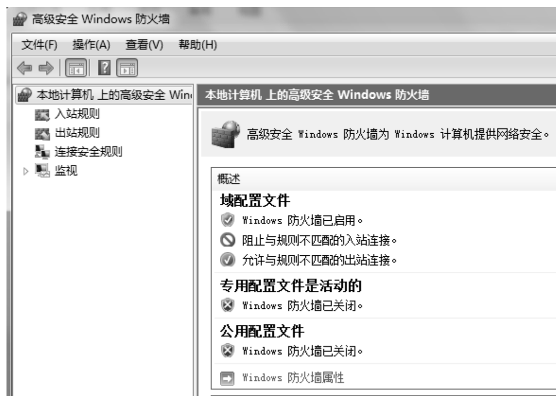


图 A-11　入站规则设置（一）



入站规则设置：新建入站规则（选择端口）→填写对应的端口号。如图 A-12、图 A-13 所示。



图 A-12　入站规则设置（二）



图 A-13　入站规则设置（三）

2．Windows 7 版

（1）安装 IIS

若使用的是 Windows 7 操作系统，系统中还未安装 IIS 服务器，可单击“开始”→“控制面板”→“程序和功能”→“打开或关闭 windows 功能”→“Internet 信息服务”，点开“Internet 信息服务”，选择“万维网服务”（还可以选择其他）然后确定即可，如图 A-14 至 A-15 所示。



图 A-14　安装 IIS（一）

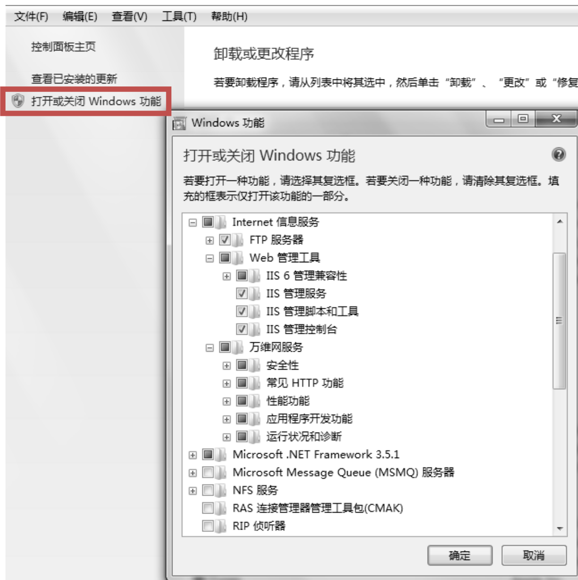


图 A-15　安装 IIS（二）

（2）启动 IIS

启动 IIS 有两种方式，一种是单击“开始”→“控制面板”→“程序和功能”→“管理工具”→“IIS 信息服务（IIS）管理器”；另一种是直接在“开始”的地址栏中输入“IIS”，然后选择“IIS 信息服务（IIS）管理器”，如图 A-16 所示。

（3）网站发布

打开 IIS 后，右击“网站”→“添加网站”，弹出网站配置窗口，如图所示。点开“选择”按钮，可以配置程序所使用的程序应用池，一般采用“ASP.NET v4.0 Classic”，也可以根据程序使用的 ASP 版本，进行相应的选择。物理目录指的是已发布的程序存在的位置。最后，可以绑定相应的 IP 地址和端口号。如图 A-17 所示。



图 A-16　启动 IIS



图 A-17　网站发布

（4）访问用户的权限设置（非必须）

确保程序及资源文件有用户访问权限，如果没有，右击要发布的程序（如 GanaSky）选择“编辑权限”，跳转到“安全”选项卡，为程序添加 Everyone 用户，且 Everyone 的访问权限至少是只读，还可以是读写权限，如图 A-18 至 A-22 所示：

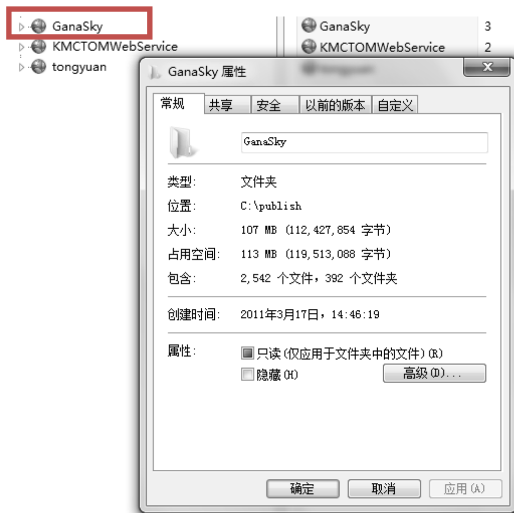


图 A-18　访问用户的权限设置（一）

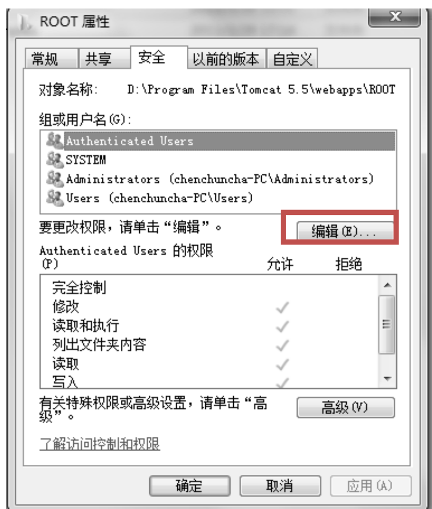


图 A-19　访问用户的权限设置（二）



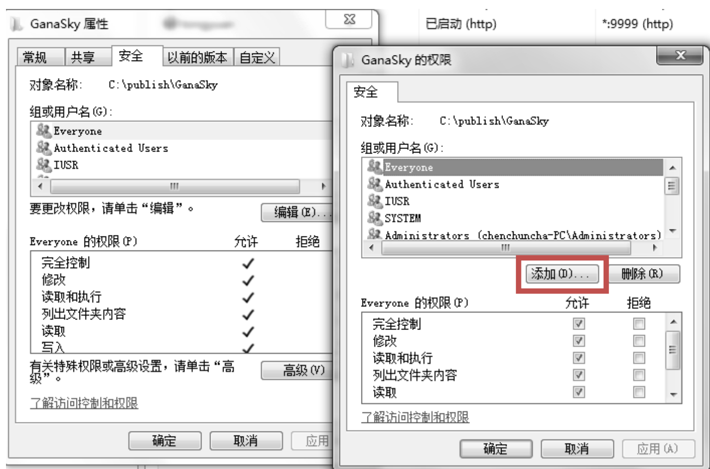


图 A-20　访问用户的权限设置（三）

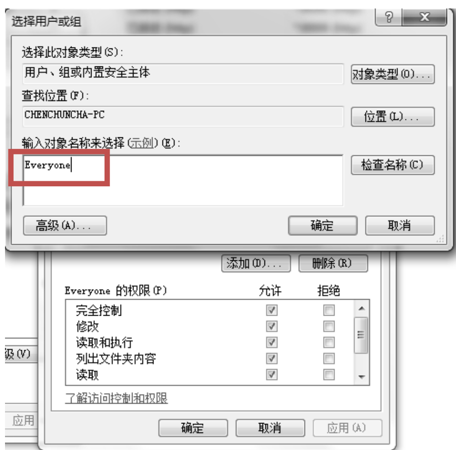


图 A-21　访问用户的权限设置（四）

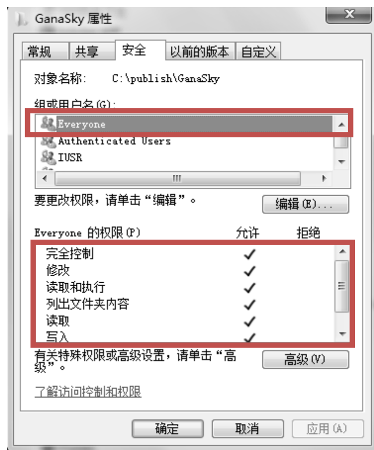


图 A-22　访问用户的权限设置（五）

（5）入站规则设置（非必须）

在高级防火墙中设置入账规则：单击“开始”→“程序”→“控制面板”→“高级防火墙”→“定义入站规则”。如图 A-23 所示。

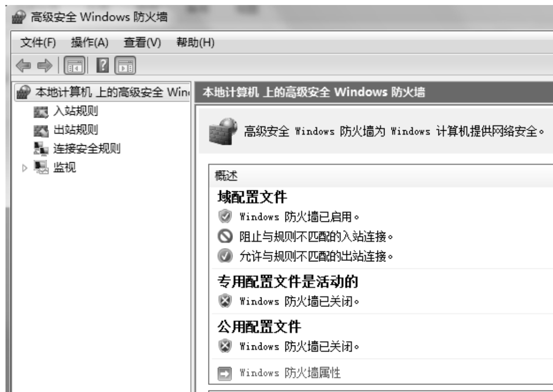


图 A-23　入站规则设置（一）

入站规则设置：新建入站规则（选择端口）→填写对应的端口号。如图 A-24、图 A-25 所示。



图 A-24　入站规则设置（二）



图 A-25　入站规则设置（三）

附录B　91 助手的安装与使用

目前市场上有许多手机连接电脑的助手，如 91 助手、豌豆荚、掌智手机助手等，91 手机助手是最常见、使用起来比较方便的一款手机连接电脑软件。本使用手册采用 91 手机助手进行导入安装同远订票系统（以下简称“订票系统”）。

91 助手的下载地址：http://soft.sj.91.com/91helper/。

91 电脑助手，显示主界面，如图 B-1。



图 B-1　91 电脑助手主界面

手机连接电脑

手机连接电脑有两种方式：USB 连接方式与 WIFI 连接方式。



USB连接

USB 连接方式是手机通过数据线与电脑进行连接，这也是最常见的连接方式。USB 连接方式操作如图 B-2、图 B-3。

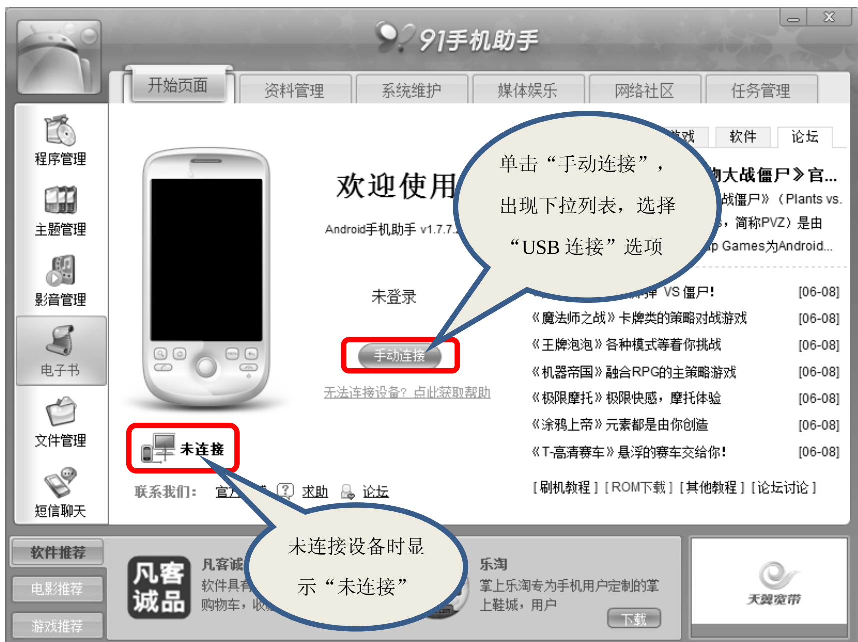


图 B-2　设置 USB 连接电脑



图 B-3　USB 连接电脑成功

WIFI连接

WIFI 连接方式是手机通过 WIFI 与电脑进行连接，这种方式一般在没有携带数据线时使用比较方便。WIFI 连接设置如图 B-4。

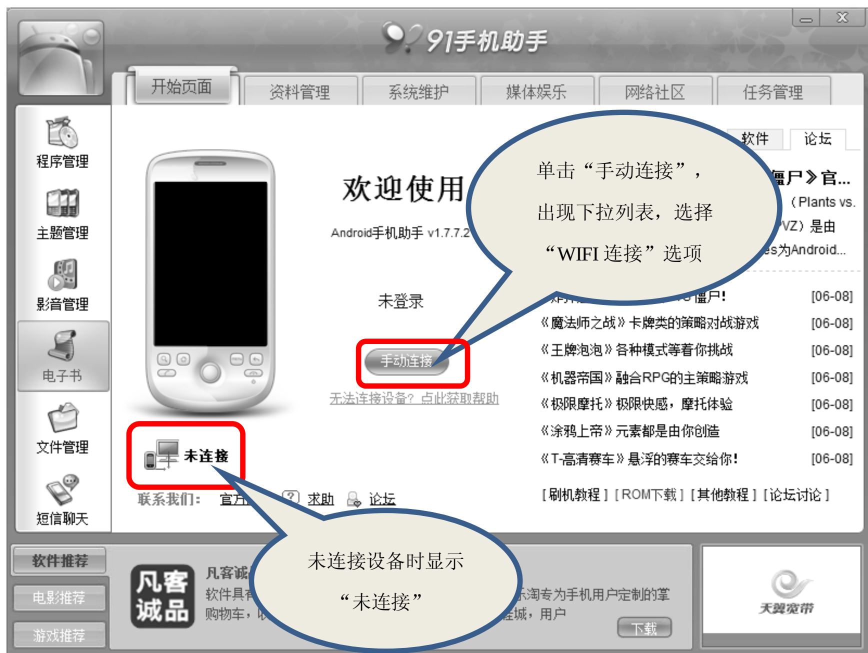


图 B-4　设置 WIFI 连接电脑

选择 WIFI 连接后，弹出电脑连接手机的设置参数，此时手机需打开 WIFI，并能保证联入互联网，获取手机 IP 地址。在图 B-5 所示的设置图中，输入手机的 IP 地址，或者手机版的 91 助手显示的验证码即可连接。



图 B-5　91 电脑助手设置相关连接参数

打开手机客户端的 91 手机助手，进入手机客户端 91 助手主界面，如图 B-6。选择界面上的“WIFI 连接 PC 端 91 手机助手”按钮。

接着，单击标识按钮后，弹出一个权限允许的提示框，如图 B-7 所示，这是保证安全连接的一种设置。选择“yes”即可。





图 B-6　91 手机助手主页

选择“yes”后，跳转到图 B-8 界面。在此界面上点击“启用 WIFI 连接”按钮，屏幕上显示了手机客户端连接电脑的验证码，该验证码即是图 B-5 中所需要输入的验证码。将此验证码输入图 B-5 中，即完成了手机客户端与电脑的连接。

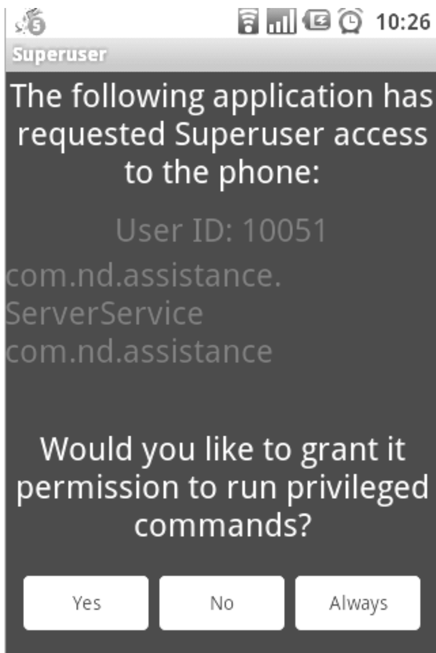


图 B-7　安全设置



图 B-8　手机客户端设置相关参数

订票系统导入手机

确定手机与电脑连接成功后，双击 91 电脑助手左侧“程序管理”选项，进入手机程序管理界面，如图 B-9。



图 B-9　91 电脑助手主界面

单击图 B-9 程序管理主界面左侧“程序管理”子项“已安装软件”选项，在右侧弹出本手机已安装的软件，如图 B-10。



图 B-10　“程序管理”界面

双击“已安装软件”选项，进行如图 B-11 的“程序列表”界面。





图 B-11 “已安装软件”界面

双击图 B-11 中“已安装软件”界面上方的“打开目录”选项，找到下载订票系统后的位置，选中并双击该系统对应的打包文件 GanaSkyFor2.1.apk（同远订票系统），91 手机助手主要将该系统导入到手机中。

手机安装订票系统

采用 USB 方式连接电脑时，利用 91 手机助手将订票系统导入到手机时，会弹出如图 B-12 中的提示窗口，点击“安装”即会自动将订票系统安装到手机上。

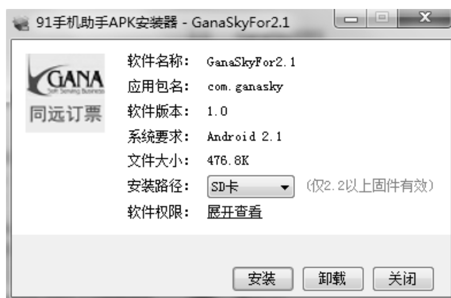


图 B-12 安装程序界面

采用 WIFI 方式连接电脑时，步骤相对麻烦。利用 91 手机助手将订票系统导入到手机时，会弹出一个提示连接的窗口，如图 B-13。IP 地址即是手机的连接互联网的 IP 地址，这里的密码要与手机端 91 助手中设置的连接密码一致。点击“连接”即可弹出如图 B-13 中的安装程序界面，点击“安装”即可完成订票系统在手机上的安装操作。

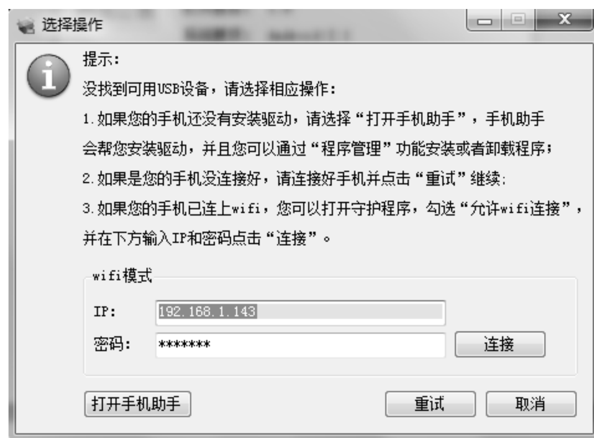


图 B-13　WIFI 连接方式下的程序安装提示

附录C　DDMS常用功能详解

DDMS 的全称是 Dalvik Debug Monitor Service，它搭起了 IDE 与手机模拟器的链接此平台，通常与模拟器配合使用，为我们提供了一个更接近真机环境的手机测试环境。提供的功能有：为测试设备截屏、针对特定的进程查看正在运行的线程以及堆信息、Logcat、广播状态信息、模拟电话呼叫、接收 SMS、虚拟地理坐标等功能。

在 Eclipse 开发平台中，DDMS 的图标是在右上角：DDMS。如果在右上角没有出现该图标，则单击菜单栏"Windows（窗口）"—>"Open Perspective（打开透视窗）"—>"DDMS"，如果还是找不到 DDMS 图标，则单击"Windows（窗口）"—>"Open Perspective（打开透视窗）"—>"Other（其他）"—>"DDMS"。如图 C-1 所示。

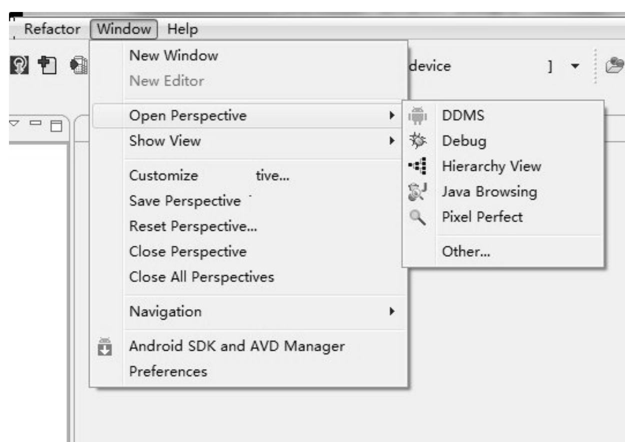


图 C-1　Eclipse 开发平台下 DDMS 位置

（1）启动 DDMS

在运行 Emulator（模拟器）状态下，DDMS 相关服务功能才奏效。因此需要启动模拟

器后，双击 DDMS 图标，DDMS 的功能才算是启动了。DDMS 对 Emulator 和外接测试机有同等效用，如果系统检测到它们（VM）同时运行，那么 DDMS 将会默认指向 Emulator。

（2）Devices

单击菜单栏中“Windows（窗口）”—>“Show View”—>“Other”—>“Android”—>“Devices”，如图 C-2、图 C-3 所示。这里可以查看到所有与 DDMS 连接的终端的详细信息以及每个终端正在运行的 APP 进程，每个进程最右边相对应的是与调试器链接的端口。因为 Android 是基于 Linux 内核开发的操作平台，同时也保留了 Linux 中特有的进程 ID，它介于进程名和端口号之间。

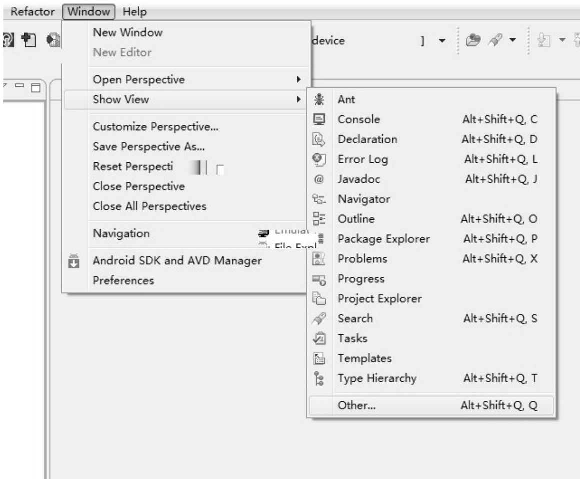


图 C-2　Eclipse 开发平台下 Devices 位置

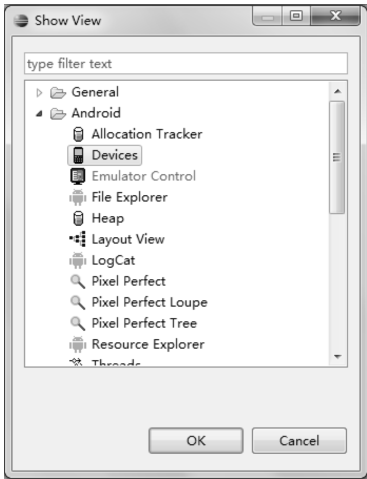


图 C-3　启动 Devices

运行模拟器，启动 DDMS，就可以在 DDMS 中看到 Devices 相关信息。如图 C-4 所示。

Name		
system_process	572	8600
android.process.acore	617	8601 / ...
com.android.phone	619	8602
android.process.media	663	8611
com.android.alarmclock	686	8613
com.example.android.snake	740	8616
com.android.inputmethod.lat	902	8603

图 C-4　DDMS 平台下的 Devices

（3）Emulator Control

单击菜单栏中“Windows（窗口）”—>“Show View”—>“Other”—>“Android”—>“Emulator Control”，如图 C-5 所示。通过这个面板的相关功能可以很容易地使测试终端模拟真实手机所具备的一些交互功能，比如：接听电话，根据选项模拟各种不同网络情况，模拟接受 SMS 消息和发送虚拟地址坐标用于测试 GPS 功能等。运行模拟器，启动 DDMS，显示 Emulator Control 相关信息。

Telephony Status：通过选项模拟语音质量以及信号连接模式。

Telephony Actions：模拟电话接听和发送 SMS 到测试终端。

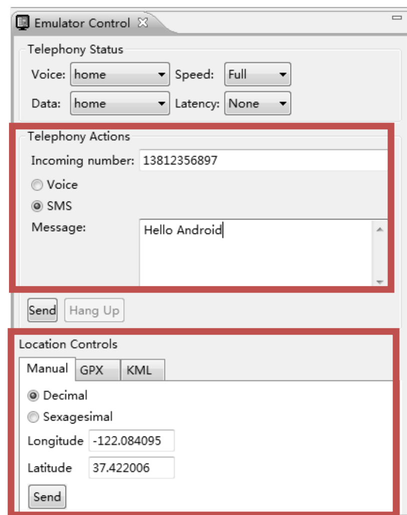


图 C-5　Eclipse 开发平台下 Emulator Control 位置

Location Control：模拟地理坐标或者模拟动态的路线坐标变化并显示预设的地理标识，可以通过以下 3 种方式：

1）Manual：手动为终端发送二维经纬坐标。

2）GPX：通过 GPX 文件导入序列动态变化地理坐标，从而模拟行进中 GPS 变化的数值。

3）KML：通过 KML 文件导入独特的地理标识，并以动态形式根据变化的地理坐标显示在测试终端。

（4）Threads、Heap、File Exporler

单击菜单栏中“Windows（窗口）”—>“Show View”—>“Other”—>“Android”—>“Threads”依次打开，在手机开发中常使用到的是 File Exporler。如图 C-6 所示。通过 File Exporler 可以查看 Android 模拟器中的文件，可以很方便地导入、导出文件。

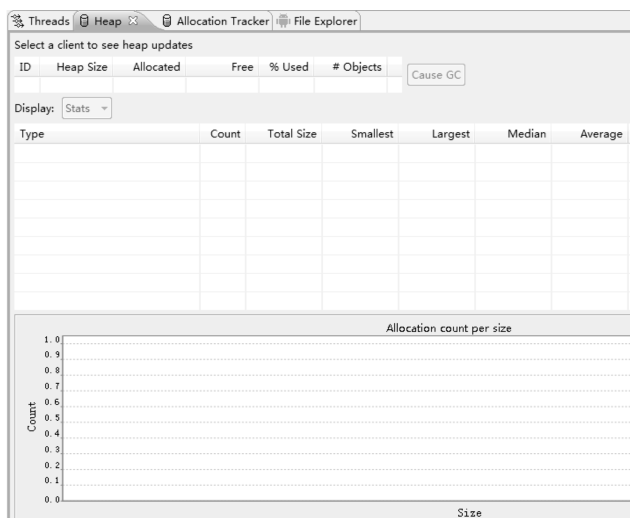


图 C-6　Eclipse 开发平台下 Threads、Heap、File Exporler 位置



如何导入、导出文件，具体步骤如下：

1）运行模拟器，打开 DDMS，在 File Explorer 找到 mnt 文件夹下的 sdcard 文件夹，如图 C-7 所示。

2）单击右上角，导入音频文件。如图 C-8 所示。

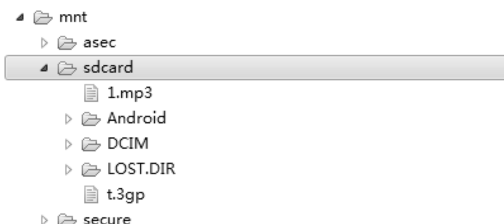


图 C-7 模拟器 DDMS 中的 sdcard 的位置



图 C-8 在模拟器中的 sdcard 中导入文件

（5）LogCat、Console

单击菜单栏中“Windows（窗口）”—>“Show View”—>“Other”—>“Android”—>“LogCat”依次打开。如图 C-9 所示。LogCat 提供日志处理功能。Android 日志系统提供了记录和查看系统调试信息的功能，日志都是从各种软件和一些系统的缓冲区中记录下来的，缓冲区可以通过 LogCat 命令来查看和使用；而 Console 是 Android 模拟器输出的信息，加载程序等信息。如图 C-10 所示。

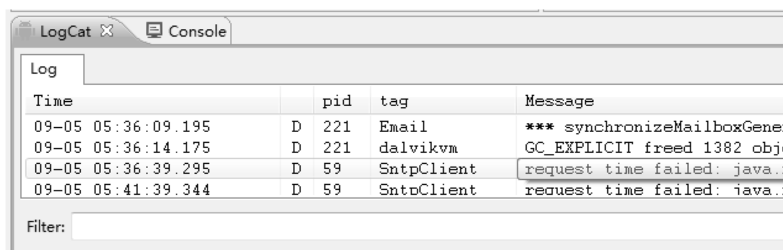


图 C-9 DDMS 平台下的 LogCat、Console



图 C-10 DDMS 平台下的 LogCat、Console

参 考 文 献

[1] 杨丰盛.Android 应用开发揭秘 [M]. 北京：机械工业出版社，2010.

[2] 百度百科[OL] .http://baike.baidu.com/.

[3] eoe 开发者门户[OL] .http://www.eoeandroid.com.

[4] ITeye 开发者网站[OL] . http://www.iteye.com/.

[5] Google 天气接口[OL] .http://www.google.com/ig/api?hl=zh_cn&weather.

[6] Google 翻译接口[OL] .http://ajax.googleapis.com/ajax/services/language/translate.



推荐图书

追逐App Store的脚步——手机软件开发者创富之路

ISBN 978-7-111-35619-6　定价：49.00元

作者：项有建

本书介绍了如何进行软件产品设计，特别是如何针对现代手机软件产品进行设计；介绍了数字产品的营销方法，特别是如何针对现代手机软件产品进行营销的方法。书中强调了用户需求以及竞争两个设计视角，介绍了"平台辐射原理"，初步解决了如何利用公式化的方法用平台推广产品的问题。

Android开发案例驱动教程

ISBN 978-7-111-35004-0　定价：69.80元

作者：关东升

本书旨在帮助读者全面掌握Android开发技术，能够实际开发Android项目。本书全面介绍了在开源的手机平台Android操作系统下的应用程序开发技术，包括UI、多线程、数据存储、多媒体、云端应用以及通信应用等方面。本书采用案例驱动模式展开讲解，既可作为高等学校的参考教材，也适合广大Android初学者和Android应用开发的程序员参考。

Windows Phone 7完美开发征程

ISBN 978-7-111-34043-0　定价：45.00元

作者：倪浩

本书以全新的Windows Phone 7手机应用程序开发为主题，采用理论和实践相结合的方法，由浅入深地讲述了新平台的基础架构、开发环境、图形图像处理、数据访问、网络通信等知识点。最后通过较为完整的实战演练，帮助读者更快地掌握项目开发的各个技术要点，使读者能够尽快投入到实际项目的开发。

从灵感到实现——打造你的第一个Windows Phone 7.5应用

ISBN 978-7-111-37452-7　定价：56.00元

作者：李永伦

本书讲述了一个真实应用的开发故事，为你呈现一个完整的开发过程，包括安装环境、需求分析、原型设计、功能开发、应用测试和错误修正。本书包含了详细的可操作的步骤，手把手教你实现相关功能，并且留下一些作业给你练手，以便巩固这节课学到的知识和技术。

从实例走进OPhone世界

ISBN 978-7-111-33030-1　定价：45.00元

作者：周轩

本书从一个开发者的角度出发，介绍了OPhone/Android系统的基础知识和开发技巧，详细讲解了无线通信、娱乐游戏、移动生活、OPhone特色应用等多种类型程序的开发流程和方法；通过介绍系统自带源代码实例，为读者提供参考资料和分析素材。环境，并配有大量插图和代码注释，为自学者提供了方便。

Qt开发Symbian应用权威指南

ISBN 978-7-111-36089-6　定价：45.00元

作者： Fitzek等　译者：DevDiv移动开发社区

本书主要是向读者介绍如何在Symbian上快速有效地创建Qt应用程序。全书共分7章，包括开发入门、Qt概述、Qt Mobility APIs、类Qt移动扩展、 Qt应用程序和Symbian本地扩展、Qt for Symbian范例。

本书可作为移动设备开发领域的初学者和专业人员的参考用书，也可作为手机开发基础课程的教材。

机工出版社·计算机分社书友会邀请卡

尊敬的读者朋友：

感谢您选择我们出版的图书！我们愿以书为媒与您做朋友！我们诚挚地邀请您加入：

“机工出版社·计算机分社书友会”

以书结缘，以书会友

加入“书友会”，您将：

★ 第一时间获知新书信息、了解作者动态；

★ 与书友们在线品书评书，谈天说地；

★ 受邀参与我社组织的各种沙龙活动，会员联谊；

★ 受邀参与我社作者和合作伙伴组织的各种技术培训和讲座；

★ 获得“书友达人”资格（积极参与互动交流活动的书友），参与每月 5 个名额的“书友试读赠阅”活动，获得最新出版精品图书 1 本。

如何加入“机工出版社·计算机分社书友会”

两步操作轻松加入书友会

Step1

访问以下任一网址：

★ 新浪官方微博：http://weibo.com/cmpjsj

★ 新浪官方博客：http://blog.sina.com.cn/cmpbookjsj

★ 腾讯官方微博：http://t.qq.com/jigongchubanshe

★ 腾讯官方博客：http://2399929378.qzone.qq.com

Step2

找到并点击调查问卷链接地址（通常位于置顶位置或公告栏），完整填写调查问卷即可。

联系方式

通信地址：北京市西城区百万庄大街 22 号
机械工业出版社计算机分社

邮政编码：100037

联系电话：010-88379750

传　　真：010-88379736

电子邮件：cmp_itbook@163.com

敬请关注我社官方微博：　http://weibo.com/cmpjsj

第一时间了解新书动态，获知书友会活动信息，与读者、作者、编辑们互动交流！

ISBN 978-7-111-37968-3
策划编辑：郝建伟
封面设计：子时文化 ZiShi Culture

在线互动交流平台
官方微博：http://weibo.com/cmpjsj
豆 瓣 网：http://site.douban.com/139085/
读者信箱：cmp_itbook@163.com

内容简介

本书从实战的角度出发，介绍了基于Android环境的Mobile/Server应用系统开发，涉及移动客户端和服务器端的开发环节，以满足智能手机在线应用的需求。全书以一个Android应用程序开发案例为主线，介绍了整个Android项目开发的过程。在内容的编排方面，为突出项目编程方法与编程思路这条主线，将Android组件的使用分配到每一个功能界面中讲解。本书涵盖了Android前端界面开发、Activity java程序的开发以及ASP后台技术开发等内容。强调了Android与后台数据通信技术、XML解析技术、数据存储技术以及与Android前端配套的ASP后台技术的实现细节。

每章的案例均有源程序，可登录www.cmpbook.com免费下载，以帮助读者深入地进行学习。

地址：北京市百万庄大街22号
邮政编码：100037
电话服务
社服中心：(010)88361066
销售一部：(010)68326294
销售二部：(010)88379649
读者购书热线：(010)88379203
网络服务
门户网：http:www.cmpbook.com
教材网：http:www.cmpedu.com
封面无防伪标均为盗版

上架建议 计算机/移动与嵌入式开发


ISBN 978-7-111-37968-3

定价：62.00元